Weighted Automata on Infinite Words in the Context of Attacker-Defender Games

V. Halava[a,b], T. Harju[a], R. Niskanen[b,*], I. Potapov[b,1]

[a] *Department of Mathematics and Statistics, University of Turku, FIN-20014 Turku, Finland*
[b] *Department of Computer Science, University of Liverpool, Ashton Building, Liverpool, L69 3BX, UK*

Abstract

The paper is devoted to several infinite-state Attacker-Defender games with reachability objectives. We prove the undecidability of checking for the existence of a winning strategy in several low-dimensional mathematical games including vector reachability games, word games and braid games. To prove these results, we consider a model of weighted automata operating on infinite words and prove that the universality problem is undecidable for this new class of weighted automata. We show that the universality problem is undecidable by using a non-standard encoding of the infinite Post correspondence problem.

Keywords: Weighted automata on infinite words, Attacker-Defender games, vector reachability, braid group, undecidability

1. Introduction

In the last decade there has been a steady, growing interest in the area of infinite-state games and computational complexity of checking whether a winning strategy exists [1, 2, 3, 4, 5, 6, 7]. Such games provide a powerful mathematical framework for a large number of computational problems. In particular, they appear in the verification, refinement and compatibility checking of reactive systems [8], analysis of programs with recursion [4], combinatorial topology and have deep connections with automata theory and logic [6, 7, 9].

In many cases of high-dimensional games, the problem of checking for the existence of a winning strategy can be computationally hard and even undecidable. Answering the same question for low-dimensional systems can also be a challenging problem. This is either due to a lack of tools for analysis of complex dynamics or due to a lack of "space" to encode directly the universal computations to show that the problem is undecidable.

*Corresponding author
Email addresses: vesa.halava@utu.fi (V. Halava), harju@utu.fi (T. Harju), r.niskanen@liverpool.ac.uk (R. Niskanen), potapov@liverpool.ac.uk (I. Potapov)

[1] The author was partially supported by EPSRC grant "Reachability problems for words, matrices and maps" (EP/M00077X/1)

In this paper we present three variants of low-dimensional Attacker-Defender games (i.e., word games, matrix games and braid games) for which it is undecidable to determine whether one of the players has a winning strategy. In addition, the proof incorporates a new language theoretical result (Theorem 2) about weighted automata on infinite words that can be efficiently used in the context of other reachability games.

An Attacker-Defender game is played in rounds, where in each round a move of Defender (Player 1) is followed by a move of Attacker (Player 2) starting from some initial position. The aim of Attacker is to reach a target position while Defender tries to keep Attacker from reaching the target position. Then, we say that Attacker has a winning strategy if she can eventually reach the target position regardless of Defender's moves. We show that in a number of restricted cases of such games, it is not possible to decide whether a winning strategy exists for a given set of moves, an initial position and a target position.

In particular, we introduce matrix games on vectors, where we show that if both players are stateless and the moves correspond to very restricted linear transformations from $SL(4, \mathbb{Z})$, the existence of a winning strategy is undecidable. There exists a simple reduction from known undecidable reachability games (robot games [10]) that leads to undecidability for a game with linear transformations in dimension six. To prove the undecidability in four-dimensional games, we first show undecidability of *word games* where players are given words over a group alphabet and in an alternating, way concatenate their words with a goal for Attacker to reach the empty word. The games on words, over semigroup alphabets, are commonly used to prove results in language theory [11, 12, 13]. We, on the other hand, define word games over group alphabets.

Later we show that it is possible to stretch the application of the proposed techniques to other models and frameworks. For example, we consider games on braids, which were recently studied in [14, 15]. Braids are classical topological objects that attracted a lot of attention due to their connections to topological knots and links, as well as their applications to polymer chemistry, molecular biology, cryptography, quantum computations and robotics [16, 17, 18, 19, 20]. In this paper we consider games on braids with only three or five strands, where the braid is modified by a composition of braids from a finite set with the target for Attacker to reach the trivial braid. We show that it is undecidable to check for the existence of a winning strategy for three strands from a given nontrivial braid and for five strands starting from the trivial braid. The reachability with a single player (i.e., with nondeterministic composition from a single set) was shown to be decidable for braids with three strands in [21].

The undecidability results of this paper are proved by using a new language-theoretic result showing that the universality problem for weighted automata $\mathcal{A}$ on infinite words is undecidable. The acceptance of an infinite word w intuitively means that there exists a finite prefix p of w such that for the word p there is a path in $\mathcal{A}$ that has zero weight. From an instance of the infinite Post correspondence problem we construct the automaton $\mathcal{A}$ that accepts all infinite words if and only if the instance does not have a solution. As the infinite Post correspondence is undecidable [22], so is the universality problem for weighted automata on infinite words.

The considered model of automaton is closely related to *integer weighted finite automata* as defined in [23] and [24], where finite automata are accepting finite words and have additive integer weights on the transitions. In [23], it was shown that the universality problem is undecidable for integer weighted finite automata on finite words by reduction

from the Post correspondence problem. In the context of a game scenario it is important to define acceptance of infinite words (which represent infinite plays in games) by considering finite prefixes reaching a target value. On the other hand, non-acceptance means that there exists an infinite computational path where none of the finite prefixes reach the target value. Then the universality for weighted automata over infinite words is the property ensuring that all infinite words are accepted (i.e., eventually reach a target in a computation path). Please note that while the universality for weighted automata on finite words implies universality for weighted automata on infinite words, the statement does not hold the other way around. Therefore the universality problem for weighted automata on infinite words is not equivalent to the universality problem for weighted automata on finite words. Our undecidability proof for weighted automata on infinite words follows the initial idea from [23] for mapping computations on words into a weighted (one-counter) automata model, which is extended with new constructions and formal proofs.

The paper is organized as follows. The next section contains basic notations and preliminaries used in the rest of the paper. In the third section we prove our main result that the universality problem is undecidable for weighted automata on infinite words. This section is the most involved and provides a new non-standard encoding of the infinite Post correspondence problem into the universality problem. Finally, in Section 4, we apply the main result to several games on mathematical objects and show that it is undecidable to check whether one of the players has a winning strategy. After each game we provide an example to illustrate the main principles of the game. We also show that the Attacker-Defender games are quite sensitive to the process of determinization. We discuss and illustrate this effect in details in the case of matrix games. Moreover, following our analysis, we formulate a problem of eventual reachability that is relevant in the context of concurrent games.

2. Notation and definitions

Words. An *infinite word* w over a finite alphabet A is an infinite sequence of letters, $w = a_0a_1a_2a_3\cdots$ where $a_i \in A$ is a letter for each $i = 0, 1, 2, \ldots$. We denote the set of all infinite words over A by A^ω. The monoid of all finite words over A is denoted by A^*. A word $u \in A^*$ is a *prefix* of $v \in A^*$, denoted by $u \leq v$, if $v = uw$ for some $w \in A^*$. If u and w are both nonempty, then the prefix u is called *proper*, denoted by $u < v$. A *prefix* of an infinite word $w \in A^\omega$ is a finite word $p \in A^*$ such that $w = pw'$ where $w' \in A^\omega$. This is also denoted by $p \leq w$. The length of a finite word u is denoted by $|u|$. By $w[i]$ we denote the ith letter of a word w, i.e., $w = w[1]w[2]\cdots$. The *reversed word* of $w = w[1]\cdots w[n]$ is denoted by $w^R = w[n]\cdots w[1]$, i.e., the order of the letters is reversed. Later in the paper the finite words will be denoted by u, v, infinite words by w and single letters by a, b, c, x, y, z.

Let $\Gamma = \{a_1, a_2, \ldots, a_n, a_1^{-1}, a_2^{-1}, \ldots, a_n^{-1}\}$ be a generating set of a free group F_Γ. The elements of F_Γ are all *reduced* words over Γ, i.e., words not containing $a_ia_i^{-1}$ or $a_i^{-1}a_i$ as a subword. In this context, we call Γ a finite *group alphabet*, i.e., an alphabet with an involution. The multiplication of two elements (reduced words) $u, v \in F_\Gamma$ corresponds to the unique reduced word of the concatenation uv. This multiplication is called *concatenation* throughout the paper. Later in the encoding of words over a

group alphabet we denote a^{-1} by $\overline{a}$ and the alphabet of inverse letters is denoted as $A^{-1} = \{a^{-1} \mid a \in A\}$.

Weighted automata. Let $\mathcal{A} = (Q, A, \sigma, q_0, F, \mathbb{Z})$ be a finite integer weighted automaton with the set of states Q, the finite alphabet A, the set of transitions $\sigma \subseteq Q \times A \times Q \times \mathbb{Z}$, the initial state q_0, the set of final states $F \subseteq Q$ and the additive group of integers[2] $\mathbb{Z}$ with identity 0, that is $(\mathbb{Z}, +, 0)$. We write the transitions in the form $t = \langle q, a, p, z\rangle \in \sigma$. In a graphical presentation a transition t is denoted by $q \xrightarrow{(a,z)} p$. Note that $\mathcal{A}$ is a non-deterministic complete automaton in a sense that for each $q \in Q$ and $a \in A$, there is at least one transition $\langle q, a, p, z\rangle \in \sigma$ for some $p \in Q$ and $z \in \mathbb{Z}$.

A finite word u is accepted by a weighted automaton if there is a computation path labeled by u such that weights of the transitions add up to 0. In [23], it was shown that the universality problem for finite words is undecidable. In order to analyse infinite runs in concurrent games, we extend the model of weighted automata to infinite words.

A *configuration* of $\mathcal{A}$ is any triple $(q, u, z) \in Q \times A^* \times \mathbb{Z}$. A configuration (q, u, z_1) is said to *yield* a configuration $(p, ua, z_1 + z_2)$ if there is a transition $\langle q, a, p, z_2\rangle \in \sigma$. This is denoted by $(q, u, z_1) \models_{\mathcal{A}} (p, ua, z_1 + z_2)$. Let $\models^*_{\mathcal{A}}$ or simply $\models^*$, if $\mathcal{A}$ is clear from the context, be the reflexive and transitive closure of the relation $\models_{\mathcal{A}}$.

Let $\pi = t_{i_0} t_{i_1} \cdots$ be an infinite path of transitions of $\mathcal{A}$, where $t_{i_j} = \langle q_{i_j}, a_{i_j}, q_{i_{j+1}}, z_j\rangle$ for $j \geq 0$ and $q_{i_0} = q_0$. We call such path π a *computation path.* Denote by $\mathcal{R}(\pi)$ the set of all reachable configurations following a path π. That is, for

$$\pi = \langle q_0, a_{i_0}, q_{i_1}, z_0\rangle \langle q_{i_1}, a_{i_1}, q_{i_2}, z_1\rangle \langle q_{i_2}, a_{i_2}, q_{i_3}, z_2\rangle \cdots ,$$

the set of reachable configurations is

$$\mathcal{R}(\pi) = \{(q_0, \varepsilon, 0), (q_{i_1}, a_{i_0}, z_0), (q_{i_2}, a_{i_0} a_{i_1}, z_0 + z_1), \\ (q_{i_3}, a_{i_0} a_{i_1} a_{i_2}, z_0 + z_1 + z_2), \ldots\}$$

Define a morphism $\|\cdot\| \colon \sigma^\omega \to A^\omega$ by setting $\|t\| = a$ if $t = \langle q, a, p, z\rangle$. Let $c = (q, u, z) \in \mathcal{R}(\pi)$ for some computation path π. The *weight* of the configuration c is $\gamma(c) = z$ and we say that c is in the state q. When a computation path π reading the word w is fixed, by the *weight of prefix* $\gamma(p)$ we denote the weight of the configuration $(q, p, z) \in \mathcal{R}(\pi)$, where $p < w$.

Let us now define the acceptance condition for weighted automata on infinite words. An infinite word $w \in A^\omega$ is accepted by $\mathcal{A}$ if there exists an infinite path π such that at least one configuration c in $\mathcal{R}(\pi)$ is in a final state and has weight $\gamma(c) = 0$. The language *accepted by* $\mathcal{A}$ is

$$L(\mathcal{A}) = \{w \in A^\omega \mid \exists \pi \in \sigma^\omega \colon ||\pi|| = w \text{ and } \exists (q, u, 0) \in \mathcal{R}(\pi) \colon q \in F\}\,.$$

Universality problem. The *universality problem* for automata over infinite words is to decide, given a weighted automaton $\mathcal{A}$, whether the language accepted by $\mathcal{A}$ is the set of all infinite words. In other words, whether or not $L(\mathcal{A}) = A^\omega$. The problem of

[2]While we restrict ourselves to the case where the weights of the automaton are elements of the additive group of integers $\mathbb{Z}$, we could define the model for any other group $(G, \cdot, \iota)$ as well.

non-universality is the complement of the universality problem, that is, whether or not $L(\mathcal{A}) \neq A^\omega$ or whether there exists a $w \in A^\omega$ such that for every computation path π of w, all configurations $(q, u, z) \in \mathcal{R}(\pi)$ in a final state do not have zero weight.

Let us compare the universality problem for automata over finite and infinite words. Let $\mathcal{B}$ be a complete weighted automaton on finite words and by $\mathcal{A}$ we refer to the same automaton on infinite words. If $\mathcal{B}$ is universal on finite words, then it is easy to see that $\mathcal{A}$ is also universal on infinite words. Indeed, if $\mathcal{B}$ accepts A^* then in particular it accepts a and all infinite words starting with a are accepted by $\mathcal{A}$. But on the other hand, if $\mathcal{B}$ is not universal on finite words, then it does not follow that $\mathcal{A}$ is non-universal on infinite words as well. Consider the weighted automaton of Figure 1 over a unary alphabet. When considering it as an automaton on infinite words, it is universal as it accepts the word a^ω. On the other hand, the automaton is not universal when operating on finite words. It is easy to see that the language accepted by the automaton is $\{a^{2n} \mid n \geq 0\}$. Thus, for acceptance conditions defined above, *the universality problem for weighted automata on finite words is not equivalent to the universality problem for weighted automata on infinite words*. Moreover, at the end of Section 3, we discuss another acceptance condition under which the universality of a weighted automaton on finite words does not imply the universality of a weighted automaton on infinite words.

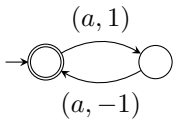

Figure 1: A weighted automaton over unary alphabet, $A = \{a\}$, which is universal over infinite words but is not universal over finite words.

Post correspondence problem. An *instance* of the *Post correspondence problem* (PCP, for short) consists of two morphisms $g, h : A^* \to B^*$ where A and B are alphabets. A nonempty word $u \in A^*$ is a *solution* of an instance (g, h) if it satisfies $g(u) = h(u)$. It is undecidable whether or not an instance of the PCP has a solution [25]. Also, the problem is undecidable for all domain alphabets A with $|A| \geq 5$ [26]. The cardinality of the domain alphabet A is said to be the *size* of the instance.

The *infinite Post correspondence problem*, ωPCP, is a natural extension of the PCP. An infinite word w is a *solution* of an instance (g, h) of the ωPCP if for every finite prefix p of w either

$$h(p) < g(p) \text{ or } g(p) < h(p). \tag{1}$$

In the ωPCP, it is asked whether a given instance has an infinite solution or not. Note that in our formulation, prefixes have to be proper. It was proven in [27] that the problem is undecidable for all domain alphabets A with $|A| \geq 9$, and it was improved to $|A| \geq 8$ in [28] for a variant of the problem where the prefixes in (1) do not have to be proper. However, it is easy to see that adding a new letter α to the alphabets and desynchronizing the morphisms g, h, gives us a solution where all prefixes have to be proper. That is, we add α to the left of each letter in the image under h, to the right of each letter in the

image under g and $g(\alpha) = a\alpha, h(\alpha) = a$ for some $a \in A$. Now the solution has to start with α, α has to appear exactly once, and the images cannot be of equal lengths because the image under g ends with α but not under h. Note that in fact, both constructions already have this desynchronizing property. See [27, 28] for more details on the morphisms g and h.

Consider an infinite word $w \in A^\omega$ that is not a solution of the ω PCP. It is clear that there exists (at least one) integer i such that $g(w)[i] \neq h(w)[i]$. That is, the letter in the position i of $g(w)$ is different from the letter in the position i of $h(w)$. We call this kind of mismatch *an error*.

Games. Attacker-Defender games are two-player zero-sum games with perfect information. Starting from some initial position, each move of Defender is followed by a move of Attacker. Attacker aims to reach a target position, while Defender tries to keep Attacker from reaching the target position. Attacker has a winning strategy if she can eventually reach the target position regardless of Defender's moves. The main computational question is to check whether Attacker has a winning strategy for a given set of moves, initial and target positions.

More formally, a game is played on an arena X, which is finite or infinite set of locations, with two special elements — an initial location x_0 and a target location x_f. Attacker has a set of moves U and Defender has a set of moves V such that for each $w \in U \cup V$ and $x \in X$, $x \cdot w \in X$. Usually, the arena is clear from the context and we do not state it explicitly. For example, in n-dimensional robot games [1, 5, 29] and n-dimensional matrix games on vectors, the arena is $X = \mathbb{Z}^n$, while in weighted word games, the arena is $X = F_\Gamma \times \mathbb{Z}$.

The game proceeds as follows: starting from x_0, first Defender chooses a move v from his set V and applies it to x_0 and then Attacker chooses a move from her set U and applies it to $x_0 \cdot v$. By repeating this process, the players create an infinite sequence of elements of X called a *play*, $\pi = (x_0, x_0v_1, x_0v_1u_1, x_0v_1u_1v_2, x_0v_1u_1v_2u_2, \ldots)$, where each $v_i \in V$ and $u_j \in U$. A play π is *winning* for Attacker if x_f appears at some position of the play. To make the special case $x_0 = x_f$ non-trivial, we require that for a play π to be winning, x_f is not the first element of π. A *strategy* of Attacker is a function $f_A : X \to U$ that tells Attacker which move to apply at each location. A strategy f_D of Defender is defined analogously. The strategy f_A is said to be a *winning strategy* for Attacker if all plays which follow f_A are winning. As a consequence of [30], Attacker-Defender games are determined, that is, Defender has a winning strategy if Attacker does not. See [31] for more details on games.

Note that the games are often considered on directed finite graphs where the vertices are partitioned between two players, and the moves are labels of the graph. In our formulation, the graph has two vertices, one for Attacker and one for Defender, no self-loops, and the edges from Attacker's (resp., Defender's) vertex to Defender's (resp., Attacker's) vertex are labeled with elements of U (resp., V).

3. The universality problem for weighted automata on infinite words

In this section we prove that the universality problem is undecidable for integer weighted automata on infinite words, by reducing the instances (of the complement) of the *infinite Post correspondence problem* to the universality problem.

Let (g, h) be a fixed instance of the ωPCP. Then $g, h\colon A^* \to B^*$ where $A = \{a_1, a_2, \ldots, a_{m-1}\}$ and $B = \{b_1, b_2, \ldots, b_{s-1}\}$. In our encoding of the ωPCP, we consider the letters of the alphabet B as natural numbers from 0 to $s-1$. We construct a non-deterministic integer weighted automaton[3] $\mathcal{A} = (Q, A, \sigma, q_0, \{q_4\}, \mathbb{Z})$, where $Q = \{q_0, q_1, q_2, q_3, q_4\}$, corresponding to the instance (g, h) such that an infinite word $w \in A^\omega$ is accepted by $\mathcal{A}$ if and only if for some finite prefix p of w, $g(p) \not\leq h(p)$ and $h(p) \not\leq g(p)$. That is, only the solutions of the ωPCP will not be accepted by $\mathcal{A}$ and if all infinite words are accepted by $\mathcal{A}$ (i.e., it is universal) then the given instance of the ωPCP does not have a solution.

In many existing reductions of the PCP or the ωPCP (for given morphisms g and h) to show undecidability of other problems, it is required to explicitly construct and store images under g and h during the simulation of the PCP or the ωPCP. In our proof, we are only going to store some partial information about the difference between the lengths of the images under morphisms g and h and some finite information that will allow us to use the non-determinism of the automaton to guess and recognize a mismatch at some position of the images.

In order to present the idea of the proof, we will first illustrate the encoding of the ωPCP into the universality problem for a weighted automaton with two weights. One weight will be used for storing the distances between different positions of letters in images under g and h, and another one to store information about a particular letter encoded as an integer. The two weights are used only to make the intuition clearer and they will be merged into a single weight by storing the second weights in the least significant digits of the first one as the second weight is only used to store finitely many values.

Let us consider an instance of the ωPCP where the image under h is always longer than the image under g and assume that the automaton reads $w \in A^\omega$. In the encoding of the ωPCP into a weighted automaton, we separate the run of the weighted automaton into five parts (A,B,C,D and E).

In part A, the automaton reads some finite prefix u of w for which it assumes that there are no errors in the images of letters of u under morphisms g, h. As there are no errors, i.e., $g(u) \leq h(u)$, it is not necessary to store the information on what the actual images are, instead the automaton, for each letter a of u, only adds the differences of lengths of the images $h(a)$ and $g(a)$ to the first weight. In part B, the automaton reads the next letter of w, x, and guesses that an error will occur at the position k in the image of $h(x)$. The automaton adds to the first weight the difference of k and the length of the image of x under g and also add $j_k = h(x)[k]$ to the second weight. Now the value of the first weight corresponds to the number of letters in the image of g between positions $|g(ux)|$ and $|h(u)| + k$; see Figure 2 for an illustration. From now on, the rest of the image of $h(w)$ starting from the position $|h(u)| + k + 1$ will not affect the acceptance of the word w. For the word w to be accepted, the letter in position $|h(u) + k$ of $h(w)$ needs to be different from the letter at the same position of $g(w)$. This happens when both weights are zero. Zero in the first weight guarantees that we consider letters of both $h(w)$ and $g(w)$ in the position $|h(u)| + k$. On the other hand, zero in the second weight guarantees that the two letters in the considered position are different. Next, in parts C and D, the automaton will check whether both conditions can be met.

[3]Note that our automaton is complete, i.e., there is a transition labeled with (a, z) from each state q_i for every $a \in A$ and some $z \in \mathbb{Z}$.

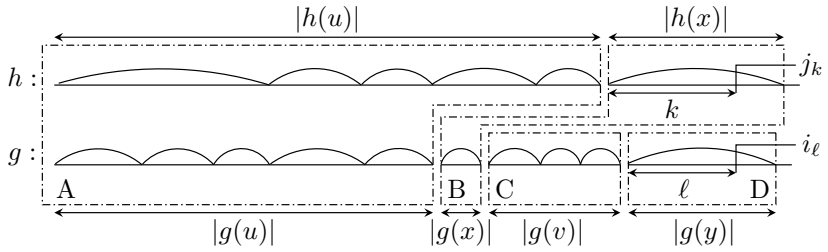

Figure 2: An illustration of a computation of the ω PCP highlighting the first four parts, A,B,C and D. Part E is not depicted.

In part C the automaton continues reading w by reading v, assuming that the letter corresponding to the erroneous position under h has not been read yet. For each letter a of v read, the length the image $g(a)$ is subtracted from the first weight. Finally, in part D, the automaton guesses that reading the next letter, y, the error will occur in the image of $g(y)$ at the position ℓ. That is, the automaton subtracts ℓ from the first weight and from the second weight, some letter $i_\ell \neq g(y)[\ell]$, i.e., any letter that is not in the ℓth position in the image of y under g. In the final part, E, the automaton is in the final state where the weights are no longer modified and the rest of the word w is read.

From the above description, in part E, the first weight is zero if the lengths of the images under h in part A, together with the position k, equal the lengths of the images under g in parts A, B and C together with the position ℓ. That is, $|h(u)|+k = |g(uxv)|+\ell$. In other words, the automaton checked whether an error occurred in the same position of $h(w)$ and $g(w)$. On the other hand, the second weight can be zero only if the letter in the image under h differs from the letter in the image under g. In other words, both weights are zero if the automaton made a correct guess that $h(w)[i] \neq g(w)[i]$ for some $i \in \mathbb{Z}$. The whole process is illustrated in Figure 3.

As mentioned previously, we can merge the two weights into a single one. The second weight is bounded and is modified only twice (once to store a letter and once to compare). By multiplying the integers stored in the first weight by s (where s is larger than the size of the image alphabet B), we create enough space to store the second weight within the first one.

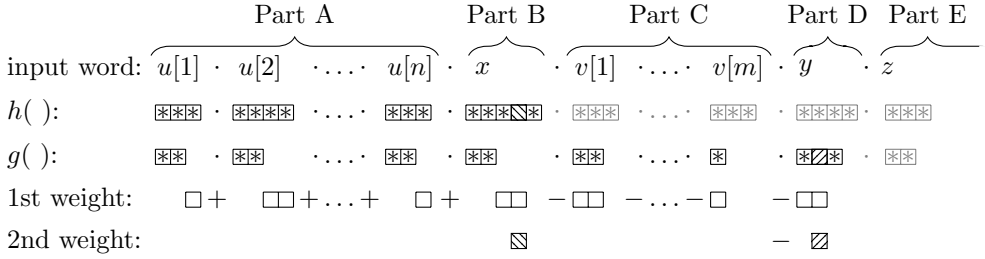

Figure 3: An illustration of a computation of the weighted automaton corresponding to an instance of the ω PCP. Here, ⊛ represents any letter of the image alphabet, while ▧ is the letter $h(x)[k]$ and ▨ is the letter $g(y)[\ell]$.

In the above description, we considered the case where the images under h were always longer than the images under g. To make the construction work for all possible cases, several computation paths need to be implemented in the automaton.

Now we are ready to formally define weighted automaton with a single weight. Let us begin with the transitions of $\mathcal{A}$. The automaton is depicted in Figure 4. Recall that the cardinality of the alphabet B is $s-1$. First for each $a \in A$, let

$$\begin{gathered}\langle q_0, a, q_1, s(|h(a)| - |g(a)|)\rangle\,, \quad \langle q_0, a, q_4, s(|h(a)| - |g(a)|)\rangle\,,\\ \langle q_1, a, q_1, s(|h(a)| - |g(a)|)\rangle\,, \quad \langle q_2, a, q_2, s(-|g(a)|)\rangle\,,\\ \langle q_3, a, q_3, s(|h(a)|)\rangle\,, \quad \langle q_1, a, q_4, 0\rangle\,, \quad \langle q_4, a, q_4, 0\rangle\end{gathered}$$

be in σ. For error checking we need the following transitions for all letters $a \in A$: Let $h(a) = b_{j_1} b_{j_2} \cdots b_{j_{n_1}}$, where $b_{j_k} \in B$, for each index $1 \le k \le n_1$, and $g(a) = b_{i_1} b_{i_2} \cdots b_{i_{n_2}}$, where $b_{i_\ell} \in B$. Then let, for each $k = 1, \ldots, n_1$ (i.e., $j_k \in \{1, \ldots, s-1\}$ for all $k = 1, \ldots, n_1$),

$$\langle q_1, a, q_2, s(k - |g(a)|) + j_k\rangle\,, \langle q_0, a, q_2, s(k - |g(a)|) + j_k\rangle \in \sigma. \tag{2}$$

For each $\ell = 1, \ldots, n_2$ and for each letter $b_c \in B$ such that $b_{i_\ell} \neq b_c \in B$, let

$$\langle q_2, a, q_4, -s\ell - c\rangle \in \sigma. \tag{3}$$

Symmetrically we define transitions for each $k = 1, \ldots, n_2$,

$$\langle q_1, a, q_3, s(-k + |h(a)|) - j_k\rangle\,, \langle q_0, a, q_3, s(-k + |h(a)|) - j_k\rangle \in \sigma, \tag{4}$$

and for each $\ell = 1, \ldots, n_1$ and for each letter $b_c \in B$ such that $b_{i_\ell} \neq b_c \in B$, let

$$\langle q_3, a, q_4, s\ell + c\rangle \in \sigma. \tag{5}$$

Next we define transitions that combine the effects of (2) and (3). For each $k = 1, \ldots, n_1$ and $\ell = 1, \ldots, n_2$ and for each letter $b_c \in B$ such that $b_{i_\ell} \neq b_c \in B$,

$$\langle q_1, a, q_4, (k - \ell)s + j_k - c\rangle \in \sigma. \tag{6}$$

Finally, for each $k = 1, \ldots, \min\{n_1, n_2\}$ and for each letter $b_c \in B$ such that $b_{j_k} \neq b_c \in B$,

$$\langle q_0, a, q_4, j_k - c\rangle \in \sigma. \tag{7}$$

We call the transitions in (2) and (4), the *error guessing transitions* and in (3) and (5), the *error verifying transitions*. Note that the transitions in (6) and (7) are both error guessing and verifying transitions.

The idea is to keep track of differences in lengths of images under g and h multiplied by s and then to guess and verify an error in the images by storing letters of the image alphabet in the least significant digits of the integer weight. The difference in lengths of the images is positive when the image under h is longer and negative when the image under g is longer. For each case there are two possibilities for position of an error. Either the difference is small enough that, after reading the next letter, there will be a position in the images where the letters differ, or the difference is large enough that the image of

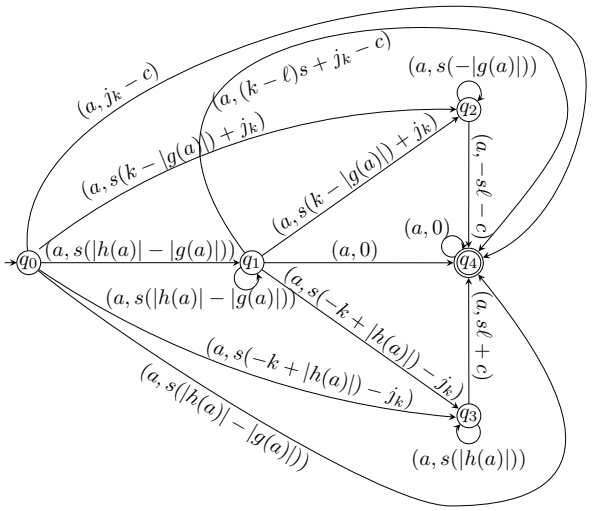

Figure 4: The weighted automaton $\mathcal{A}$. In the figure $a \in A$.

the second morphism has to catch up before the error can be verified. Also, from our formulation of the ω PCP, it is possible that the images are of equal length which means that the word is not a solution of the ω PCP.

The following lemma shows that for each case, there exists a path with zero weight ending in the state q_4.

Lemma 1. *Let $w \in A^\omega$. Then w is a solution of an instance (g, h) of the ω PCP if and only if $w \notin L(\mathcal{A})$.*

Proof. Assume first that w is a solution of the instance (g, h) of the ω PCP and assume, contrary to the claim, that there is an accepting path reading w in $\mathcal{A}$.

We consider state transitions of the accepting path. Clearly the accepting path has to visit the states q_0 and q_4. There are five cases to be considered.

(i) No other states are visited, i.e., a transition from q_0 to q_4 is used, or

(ii) the states q_1 and q_2 are visited, i.e., transitions from (2) and (3) are used, or

(iii) the states q_1 and q_3 are visited, i.e., transitions from (4) and (5) are used, or

(iv) only the state q_2 or q_3 is visited, i.e., a transition from q_0 to q_2 or to q_3 is used, or

(v) the state q_1 is visited but not q_2 nor q_3.

Assume first that w is accepted by a path π that goes to q_4 directly from q_0. To get zero weight, either $|h(w[1])| = |g(w[1])|$, meaning that w is not a solution, or $j_k - c = 0$ for some position k, but this is not possible because the letters at the position k are equal under both morphisms.

Consider the second case, where the path visits q_2. In other words, w has a prefix $p = auxvy$, where $a, x, y \in A$ and $u, v \in A^*$, such that a is read using the transition $\langle q_0, a, q_1, s(|h(a)| - |g(a)|)\rangle$, u is read in the state q_1, when reading the letter x the state changes to q_2, and v is read in the state q_2, and when reading the letter y the state changes to q_4. The weight $\gamma(p)$ of p is now

$$\begin{aligned} &s(|h(au)| - |g(au)|) + s(k - |g(x)|) + j_k + s(-|g(v)|) + (-s\ell - c) \\ &\qquad = s\big(|h(au)| + k - |g(auxv)| - \ell\big) + j_k - c \end{aligned}$$

where $h(x)[k] = b_{j_k}$ and $g(y)[\ell] \neq b_c$. As $j_k < s$ and $c < s$, we have that $\gamma(p) = 0$ if and only if $|h(au)| + k = |g(auxv)| + \ell$ and $j_k = c$. Denote $r = |h(au)| + k$. Now, $\gamma(p) = 0$ if and only if $h(w)[r] = b_{j_k} \neq b_c = g(w)[r]$, which is a contradiction since w was assumed to be a solution of (g, h).

The third case is proved analogously.

In the fourth case there are two symmetric subcases. Assume first that the path visits q_2, then the word w has a prefix $p = xvy$, where $x, y \in A$ and $v \in A^*$, such that x is read using the transition $\langle q_0, x, q_2, s(k - |g(x)|) + j_k\rangle$, v is read in the state q_2 and y is read using the transition $\langle q_2, y, q_4, -s\ell - c\rangle$ where $h(x)[k] = b_{j_k}$ and $g(y)[\ell] \neq b_c$. Now

$$\gamma(p) = s(k - |g(x)|) + j_k - s(|g(v)|) - s\ell - c.$$

Since this path is accepting, the weight has to be zero. That is, $h(w)[k] \neq g(w)[|g(xv)| + \ell]$ and $k = |g(xv)| + \ell$. But this is not possible because w is a solution. The proof of the subcase, where the path visits q_3, is symmetric.

In the final case, where the accepting path visits q_1 but does not visit q_2 nor q_3, there are two possible paths. Either the final transition is $\langle q_1, a, q_4, 0\rangle$ which means that for some prefix p, $|h(p)| = |g(p)|$, i.e., w is not a solution. The other possibility is that the final transition is $\langle q_1, a, q_4, (k - \ell)s + j_k - c\rangle$. In this case there is a prefix $p = aux$, where $a, x \in A$ and $u \in A^*$, such that a is read using the transition $\langle q_0, a, q_1, s(|h(a)| - |g(a)|)\rangle$, u is read in q_1, and x is read using the transition $\langle q_1, x, q_4, s(k - \ell) + j_k - c\rangle$. The weight $\gamma(p)$ is now

$$s(|h(au)| + k - |g(au)| - \ell) + j_k - c = 0$$

if and only if $|h(au)| + k = |g(au)| + \ell$ and $j_k = c$. Again, this is not possible since w is a solution.

For the second half of the claim, assume that w is not a solution. We summarize the possible cases for a word $w \in A^\omega$ that is not a solution of the ω PCP. For w, there exists a prefix p such that at least one of the following cases holds:

(a) $|h(p)| = |g(p)|$ and $|p| = 1$, or

(b) $|h(p)| \neq |g(p)|$, $h(p)[k] \neq g(p)[k]$ and this error is in the image of the first letter of p under both h and g, or

(c) $|h(p)| > |g(p)|$ and an error is in the images of different letters of p under h and g, or

(d) $|g(p)| > |h(p)|$ and an error is in the images of different letters of p under h and g, or

(e) $|h(p)| \neq |g(p)|$, $h(p)[k] \neq g(p)[k]$ and this error is in the image of the same letter of p, but not the first, under both h and g, or

(f) $|h(p)| = |g(p)|$ and $|p| > 1$.

Intuitively, the cases (a) and (b) have the same path as the case (i), the case (c) as (ii) and (iv), the case (d) as (iii) and (iv), and the cases (e) and (f) as the case (v).

Let us consider the first case. Now the prefix $p = a$ and w is accepted by using the transition $\langle q_0, a, q_4, s(|h(a)| - |g(a)|)\rangle = \langle q_0, a, q_4, 0\rangle$.

Now consider the second case. Let the first letter of p be a. By using the transition $\langle q_0, a, q_4, j_k - c\rangle$, where $j_k = c$, we get an accepting computation for w.

Let us consider the third case. There are two subcases for this case. Either state q_1 is visited, or not. Assume first that q_1 is visited. Let r be the minimal position for which $h(w)[r] \neq g(w)[r]$. In other words, for $p = c_1 \cdots c_n$, there exists a position $t < n$ such that $r = |h(c_1 c_2 \cdots c_{t-1})| + k$ where $k \leq |h(c_t)|$, and $r = |g(c_1 c_2 \cdots c_{n-1})| + \ell$ where $\ell \leq |g(c_n)|$. Denote $h(w)[r] = b_{j_k}$. It is the kth letter of the image $h(c_t)$, and $g(w)[r]$ is the ℓth letter of the image $g(c_n)$. Also, these letters are nonequal.

Now, w is accepted in the state q_4 with the following path: First c_1 is read with the transition $\langle q_0, c_1, q_1, s(|h(c_1)| - |g(c_1)|)\rangle$, and the prefix $c_2 \cdots c_{t-1}$ is read in the state q_1 with weight $s(|h(c_2 \cdots c_{t-1})| - |g(c_2 \cdots c_{t-1})|)$. When reading c_t, the automaton uses the error guessing transition $\langle q_1, c_t, q_2, s(k - |g(c_t)|) + j_k\rangle$, and then the word $c_{t+1} \cdots c_{n-1}$ is read in the state q_2 with weight $s(-|g(c_{t+1} \cdots c_{n-1})|)$. Finally, while reading c_n, the state q_4 is reached by the error verifying transition $\langle q_2, c_n, q_4, -s\ell - j_k\rangle$. Note that such an error verifying transition exists as the ℓth letter in $g(c_n)$ is not equal to the kth letter b_{j_k} of $h(c_t)$. Naturally after reaching q_4, the weight does not change, as for all letters there are only transitions with zero weight. Now the weight of the above path is

$$\begin{aligned} s(|h(c_1 \cdots c_{t-1})| - |g(c_1 \cdots c_{t-1})|) + s(k - |g(c_t)|) \\ + j_k + s(-|g(c_{t+1} \cdots c_{n-1})|) - s\ell - j_k \\ = s\big(|h(c_1 \cdots c_{t-1})| + k - |g(c_1 \cdots c_{n-1})| - \ell\big) \\ = s(r - r) = 0. \end{aligned}$$

Therefore, w is accepted as claimed. The proof of the second subcase, where the state q_1 is not visited, is analogous.

The fourth case is symmetric to the third and is proven in the similar manner.

Assume then that the fifth case holds. Let $p = aub$ where $a, b \in A$ and $u \in A^*$. Using the transition $\langle q_0, a, q_1, s(|h(a)| - |g(a)|)\rangle$ followed by the transition

$$\langle q_1, u[i], q_1, s(|h(u[i])| - |g(u[i])|)\rangle$$

for each letter $u[i]$ of u and finally the transition $\langle q_1, b, q_4, (k - \ell)s + j_k - c\rangle$, where $j_k = c$, the computation reaches q_4. The weight is

$$s(|h(au)| - |g(au)|) + s(k - \ell) + j_k - j_k = 0$$

when k and ℓ are according to be the position of an error in both images.

Finally, assume that the sixth case holds. Now consider $p = au$ where $a \in A$ and $u \in A^*$. Using the transition $\langle q_0, a, q_1, s(|h(a)| - |g(a)|)\rangle$ followed by the transition

$$\langle q_1, u[i], q_1, s(|h(u[i])| - |g(u[i])|)\rangle$$

for each letter $u[i]$ of u the automaton reads p. Finally using the transition $\langle q_1, b, q_4, 0\rangle$, where $b = w[|p| + 1]$, the computation reaches q_4. By our assumption $|h(p)| = |g(p)|$ and thus the total weight is 0. □

Theorem 2. *It is undecidable whether or not $L(\mathcal{A}) = A^\omega$ holds for some 5-state integer weighted automata $\mathcal{A}$ over its alphabet A.*

Proof. The claim follows from Lemma 1 and the undecidability of the infinite PCP [22]. The automaton $\mathcal{A}$ is depicted in Figure 4. □

Corollary 3. *It is undecidable whether or not for a weighted automaton $\mathcal{A}$, there exists a word $w \in A^\omega$ such that for every computation path π of w, all configurations $(q, u, z) \in \mathcal{R}(\pi)$ in a final state do not have zero weight.*

Proof. The statement formulates the condition for non-universality. By the previous theorem, the universality problem is undecidable, and thus so, is its complement problem. □

Example 4. Let $A = \{a, b, c, d\}$ and $B = \{1, 2\}$. Consider morphisms $g, h : A^* \to B^*$ defined by

$$\begin{array}{lllll} g: & g(a) = 1, & g(b) = 2, & g(c) = 21, & g(d) = 1, \\ h: & h(a) = 1, & h(b) = 12, & h(c) = 2, & h(d) = 12. \end{array}$$

The corresponding automaton has 72 transitions and is not presented here.

Consider an infinite word starting with b. Clearly it is not a solution of the ωPCP. This word is accepted by the automaton by using the transition $\langle q_0, b, q_4, 0\rangle$, which takes automaton to the final state with weight zero. Consider an infinite word w beginning with *dbdbaca*. It is easy to see that w is not a solution either since

$$g(dbdbaca) = 12121211 \not< 12121212121 = h(dbdbaca).$$

For the automaton to accept this word, we have to nondeterministically guess that the error will occur in the second letter of the image of the final b and verify that it does occur. The corresponding computation path starts with

$$\begin{aligned} &\langle q_0, d, q_1, 3\rangle \langle q_1, b, q_1, 3\rangle \langle q_1, d, q_1, 3\rangle \langle q_1, b, q_2, 5\rangle \\ &\quad \langle q_2, a, q_2, -3\rangle \langle q_2, c, q_2, -6\rangle \langle q_2, a, q_4, -5\rangle \end{aligned}$$

and the weight is $3 + 3 + 3 + 5 - 3 - 6 - 5 = 0$, that is, the word is accepted.

Now consider the word $caa\cdots = ca^\omega$. It is easy to see that it is a solution of our instance of the ωPCP. We will show that no path starting from q_0 and ending in q_4 has weight 0. Paths starting with transitions $\langle q_0, c, q_4, 1\rangle$ and $\langle q_0, c, q_4, -3\rangle$ have non-zero weights. The other paths not visiting q_3 have negative weights, so the computation has to visit q_3. But in that case the path is either

$$\begin{aligned} &\langle q_0, c, q_3, -3\rangle \langle q_3, a, q_3, 3\rangle^k \langle q_3, a, q_4, 5\rangle \langle q_4, a, q_4, 0\rangle^\omega \quad \text{or} \\ &\langle q_0, c, q_1, -3\rangle \langle q_1, a, q_1, 0\rangle^n \langle q_1, a, q_3, -1\rangle \langle q_3, a, q_3, 3\rangle^m \langle q_3, a, q_4, 5\rangle \langle q_4, a, q_4, 0\rangle^\omega \end{aligned}$$

for some $k, n, m \in \mathbb{N}$. The weights are respectively either $-3 + 3k + 5 \neq 0$ or $-3 + 0n - 1 + 3m + 5 \neq 0$.

The result of Theorem 2 also holds for a more restricted acceptance condition, similar to that of Büchi automata, where an infinite word is accepted if and only if there is a computation path with an infinite number of prefixes with weight zero. This is clear from the construction of the automaton. If the state q_4 is reached with zero weight, then it remains unchanged when additional letters are read. For this acceptance condition the universality of a weighted automaton on finite words does not imply the universality for a weighted automaton on infinite words. However, for simplicity, we use the original acceptance condition in the proofs in Section 4.

4. Applications to Attacker-Defender games

In this section we provide a number of applications for our new result on the universality of weighted automata on infinite words. We connect the idea of a one-weight computation with games on different mathematical objects such as words, matrices, vectors and braids.

Following the result for the weighted automata on infinite words (Theorem 2) we can now define a simple scenario of an undecidable infinite-state game that can be also applied to other game frameworks. Let $\mathcal{A}$ be a weighted automaton. In the game, Defender will play an input word letter-by-letter and Attacker will simulate a run on $\mathcal{A}$ using the letters provided by Defender. In other words, Attacker has to verify whether the provided word is accepted by $\mathcal{A}$.

In the above framework Defender will have a winning strategy if there is a solution for the infinite PCP and Attacker will have a winning strategy otherwise.

4.1. Weighted word games

Let us define the Attacker-Defender game on words where the moves of Attacker and Defender correspond to concatenations of words (over free group alphabet). This game will allow us to prove nontrivial results for games with *low-dimensional* linear transformations and topological objects just by using injective homomorphism (i.e., monomorphism) to map words into other mathematical objects.

A *weighted word game* consists of two players, *Attacker* and *Defender* having sets of words $\{u_1, \ldots, u_r\} \subseteq F_\Gamma$ and $\{v_1, \ldots, v_s\} \subseteq F_\Gamma$ respectively, where Γ is a finite group alphabet, and integers $x_{u_1}, \ldots, x_{u_r}, x_{v_1}, \ldots, x_{v_s}$ corresponding to each word. That is,

$$U = \{(u_1, x_{u_1}), \ldots, (u_r, x_{u_r})\}$$

and

$$V = \{(v_1, x_{v_1}), \ldots, (v_s, x_{v_s})\}.$$

An initial position is the pair $(w, 0)$, where $w \in F_\Gamma$ and 0 is the initial value of the weight, and the target position of this game is the group identity, i.e., the empty word, with zero weight. A *configuration* of a game at time t is denoted by a word w_t and integer x as a weight. In each round of the game both Defender and Attacker concatenate their words and update the weight. Clearly $w_t = w \cdot v_{i_1} \cdot u_{i_1} \cdot v_{i_2} \cdot u_{i_2} \cdot \ldots \cdot v_{i_t} \cdot u_{i_t}$ after t rounds of the game, where u_{i_j} and v_{i_j} are words from the sets U and V respectively, and the weight is $\sum_{j=1}^{t}(x_{v_{i_j}} + x_{u_{i_j}})$. The decision problem for the word game is to check whether there exists a winning strategy for Attacker to reach the empty word with zero weight.

Before proving the main theorem, we consider two auxiliary results. In the first lemma, we present an encoding from arbitrary group alphabet to binary group alphabet. In the second lemma, we modify the automaton of Theorem 2 in order to remove loops.

Lemma 5. *Let $\Sigma' = \{z_1, \ldots, z_\ell, \overline{z_1}, \ldots, \overline{z_\ell}\}$ be a group alphabet and $\Sigma_2 = \{c, d, \overline{c}, \overline{d}\}$ be a binary group alphabet. Define the mapping $\alpha : \Sigma' \to \Sigma_2^*$ by: $\alpha(z_i) = c^i d \overline{c}^i, \alpha(\overline{z_i}) = c^i \overline{d} \overline{c}^i$, where $1 \leq i \leq \ell$. Then α is a monomorphism. Note that α can be extended to domain Σ'^* in the usual way [32, 33].*

Lemma 6. *It is undecidable whether or not $L(\mathcal{B}) = A^\omega$ holds for 9-state integer weighted automata without self-loops over an alphabet A.*

Proof. Let $\mathcal{A} = (\{q_0, q_1, q_2, q_3, q_4\}, A, \sigma, q_0, \{q_4\}, \mathbb{Z})$ be the automaton of Theorem 2. We construct an automaton $\mathcal{B} = (Q_\mathcal{B}, A, \sigma', q_0, \{q_4\}, \mathbb{Z})$ where

$$Q_\mathcal{B} = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8\}.$$

The idea is to have two copies of $\mathcal{A}$ and, instead of self-loops, we switch between the copies. In $\mathcal{B}$, the set of transitions σ' is defined as follows:

$$\begin{aligned}\sigma' = &\{\langle q_0, a, q, z\rangle \in \sigma\} \\ &\cup \{\langle q_i, a, q_j, z\rangle, \langle q_{i+4}, a, q_{j+4}, z\rangle \mid \langle q_i, a, q_j, z\rangle \in \sigma, i \neq 0, i \neq j\} \\ &\cup \{\langle q_i, a, q_{i+4}, z\rangle, \langle q_{i+4}, a, q_i, z\rangle \mid \langle q_i, a, q_i, z\rangle \in \sigma\}.\end{aligned}$$

It is easy to see that both automata accept the same language. □

We are now ready to prove the first result on Attacker-Defender games.

Theorem 7. *It is undecidable whether Attacker has a winning strategy to reach the target position $(\varepsilon, 0)$ from a given initial position $(w, 0)$, where $w \in F_\Gamma$, in the weighted word game with words over a binary group alphabet.*

Proof. The proof is based on the reduction of the universality problem for weighted automata on infinite words to the problem of checking for the existence of a winning strategy in the weighted word game. Let $\mathcal{B} = (Q_\mathcal{B}, A, \sigma', q_0, \{q_4\}, \mathbb{Z})$ from Lemma 6.

Let us define the following initial instance of the weighted word game. Defender's words are just single letters from the alphabet A with weight 0. That is, $V = \{(a, 0) \mid a \in A\}$. Attacker has three types of moves. Either she appends the dummy letter #, begins the simulation of $\mathcal{B}$, or continues the simulation. More precisely, Attacker's words are over the group alphabet $\Gamma = A \cup A^{-1} \cup Q_\mathcal{B} \cup Q_B^{-1} \cup \{\#, \overline{\#}\}$ and

$$\begin{aligned}U = &\{(\#, 0)\} \cup \{(\overline{a}\,\overline{q_i}, z) \mid \langle q_i, a, q_4, z\rangle \in \sigma'\} \\ &\cup \{(\overline{b} q_j \overline{\#}\,\overline{a}\,\overline{q_i}, z) \mid \langle q_i, a, q_j, z\rangle \in \sigma', b \in A\}.\end{aligned}$$

The initial position is $(q_4\#, 0)$ and the target position $(q_4\#\overline{q_0}, 0)$. Then in the above game Defender can avoid reaching the configuration $(q_4\#\overline{q_0}, 0)$ if and only if there is an infinite word that is not accepted by the weighted automaton $\mathcal{B}$. Note that Attacker is simulating $\mathcal{B}$ in reverse, from the final state q_4 to the initial state q_0 following all edges of $\mathcal{B}$ in the opposite direction.

Let us assume that Defender has played a word p, where $|p| = n$, and Attacker decides to simulate $\mathcal{B}$, that is, she wants to show that $pw \in L(\mathcal{B})$ for any $w \in A^\omega$. Prior to this decision Attacker had played only # and so the current configuration is

$$(q_4\# \cdot p[1] \cdot \# \cdot p[2] \cdot \ldots \cdot \# \cdot p[n], 0).$$

Let π be a computation path such that $(q, p, z) \in \mathcal{R}(\pi)$ for some $q \in Q_\mathcal{B}$ and $z \in \mathbb{Z}$. First, we assume that Attacker follows π and show that the target word is reachable if $(q_4, p, 0) \in \mathcal{R}(\pi)$, then we show that the target word is not reachable if Attacker does not follow π. Let $(q_{i_1}, p[1], q_{i_2}, z_1), \ldots, (q_{i_n}, p[n], q_4, z_n)$ be the n elements of π before the computation reaches the final state. After playing moves corresponding to these transitions, the configuration is

$$\begin{bmatrix} q_4\# \\ 0 \end{bmatrix} \underset{D}{\begin{bmatrix} p[1] \\ 0 \end{bmatrix}} \underset{A}{\begin{bmatrix} \# \\ 0 \end{bmatrix}} \cdots \underset{D}{\begin{bmatrix} p[n] \\ 0 \end{bmatrix}} \underset{A}{\begin{bmatrix} \overline{p[n]}\,\overline{q_{i_n}} \\ z_n \end{bmatrix}} \underset{D}{\begin{bmatrix} a_{j_1} \\ 0 \end{bmatrix}} \underset{A}{\begin{bmatrix} \overline{a_{j_1}} q_{i_n} \overline{\#}\,\overline{p[n-1]}\,\overline{q_{i_{n-1}}} \\ z_{n-1} \end{bmatrix}} \cdots \underset{D}{\begin{bmatrix} a_{j_n} \\ 0 \end{bmatrix}} \underset{A}{\begin{bmatrix} \overline{a_{j_n}} q_{i_1} \overline{\#}\,\overline{p[1]}\,\overline{q_{i_1}} \\ z_1 \end{bmatrix}},$$

where the letter D indicates a move of Defender and the letter A indicates a move of Attacker. In the first component, we have the reduced word $q_4\#\overline{q_{i_1}}$ and in the second component we have weight $\sum_{i=1}^{n} z_i = z$. Note that any letter a_{j_k}, which are played by Defender after Attacker has started the simulation of the automaton, plays no role in the computation as it is immediately canceled by Attacker. If the configuration $(q_4, p, 0)$ is reachable, then $q_{i_1} = q_0$ and $z = 0$, so the configuration is $(q_4\#\overline{q_{i_1}}, 0)$ and Attacker wins. On the other hand, if the configuration $(q_4, p, 0)$ is not reachable, then either $q_{i_1} \neq q_0$ or $z \neq 0$. This means that the current configuration is not the target configuration, therefore Attacker has not won (yet).

Next we show that Attacker does not have a winning strategy if she does not follow a computation path of automaton $\mathcal{B}$. We focus on the first component of the game. There are three (not mutually exclusive) cases to consider when Attacker does not follow a computation path. Attacker can play a move such that, in the reduced word,

- there are at least two inverse letters corresponding to the states of $\mathcal{B}$, or
- there is an inverse letter $\overline{a}$, or
- there is an inverse letter corresponding to a state of $\mathcal{B}$ followed by #.

In the first case Attacker does not have a winning strategy because the available moves of Attacker do not decrease the number of inverse letters corresponding to the states. The second case is also straightforward. Attacker's moves do not contain letters $a \in A$, so only Defender can cancel the inverse letter. Therefore, Defender will play $b \neq a$, ensuring that the reduced word $q_4\#\overline{q_0}$ cannot be reached. In the final case, after Attacker playing $(\#, 0)$ and Defender playing $a \in A$, the reduced word is of the form $q_4\#w\overline{q_i}a'\#a$. Attacker cannot reduce the length of this word as the only moves that cancel the final letter a contain an inverse letter corresponding to the states, after which the reduced word contains two inverse letters corresponding to the states of $\mathcal{B}$ and Attacker does not have a winning strategy as shown in the first case.

We have analysed all the possible ways Attacker can deviate from a faithful simulation of $\mathcal{B}$ and showed that Defender has a winning strategy in them. That is, Attacker has a winning strategy if and only if $\mathcal{B}$ is universal.

In order to get a game where the word of a winning configuration is the empty word, rather than $q_4\#\overline{q_0}$, we need to have an extra move for Attacker, and to make sure that no false solutions are added. The simple construction of adding words $\overline{a}q_0\overline{\#}\overline{q_4}$ for all $a \in A$ creates no new solutions as there is no way to reach ε, after $q_4\#$ has been canceled out and this is the only way to cancel $q_4\#$.

In order to complete the proof, we will require the encoding of Lemma 5 between words over an arbitrary group alphabet and a binary group alphabet. The lemma's morphism gives a way to map words from an arbitrary sized group alphabet into the set of words over a free group alphabet with only two letters. □

Example 8. Let $\mathcal{A}$ be a weighted automaton depicted in Figure 5 where q_0 is the initial state and q_3 is the final state. We construct the corresponding weighted word game. To keep the example clearer, we refrain from doing the final step of the proof. That is, instead of a binary group alphabet, we use a larger group alphabet. Defender has two moves $(a, 0), (b, 0)$ and Attacker has the following set of 28 moves

$$\left\{ \begin{bmatrix} \# \\ 0 \end{bmatrix}, \begin{bmatrix} \overline{a}\,\overline{q_2} \\ 1 \end{bmatrix}, \begin{bmatrix} \overline{a}\,\overline{q_2} \\ -2 \end{bmatrix}, \begin{bmatrix} \overline{b}\,\overline{q_2} \\ 0 \end{bmatrix}, \begin{bmatrix} \overline{a}\,\overline{q_1} \\ 0 \end{bmatrix}, \begin{bmatrix} \overline{b}\,\overline{q_1} \\ -1 \end{bmatrix}, \begin{bmatrix} \overline{a}\,\overline{q_0} \\ 2 \end{bmatrix}, \begin{bmatrix} \overline{b}\,\overline{q_0} \\ 0 \end{bmatrix}, \begin{bmatrix} \overline{a}q_3\overline{\#}\overline{a}\,\overline{q_2} \\ -2 \end{bmatrix}, \begin{bmatrix} \overline{b}q_3\overline{\#}\overline{a}\,\overline{q_2} \\ -2 \end{bmatrix}, \right.$$

$$\begin{bmatrix} \overline{a}q_3\overline{\#}\overline{b}\,\overline{q_2} \\ 0 \end{bmatrix}, \begin{bmatrix} \overline{b}q_3\overline{\#}\overline{b}\,\overline{q_2} \\ 0 \end{bmatrix}, \begin{bmatrix} \overline{a}q_2\overline{\#}\overline{a}\,\overline{q_3} \\ 0 \end{bmatrix}, \begin{bmatrix} \overline{b}q_2\overline{\#}\overline{a}\,\overline{q_3} \\ 0 \end{bmatrix}, \begin{bmatrix} \overline{a}q_2\overline{\#}\overline{b}\,\overline{q_3} \\ 0 \end{bmatrix}, \begin{bmatrix} \overline{b}q_2\overline{\#}\overline{b}\,\overline{q_3} \\ 0 \end{bmatrix},$$

$$\begin{bmatrix} \overline{a}q_3\overline{\#}\overline{a}\,\overline{q_1} \\ 0 \end{bmatrix}, \begin{bmatrix} \overline{b}q_3\overline{\#}\overline{a}\,\overline{q_1} \\ 0 \end{bmatrix}, \begin{bmatrix} \overline{a}q_3\overline{\#}\overline{b}\,\overline{q_1} \\ -1 \end{bmatrix}, \begin{bmatrix} \overline{b}q_3\overline{\#}\overline{b}\,\overline{q_1} \\ -1 \end{bmatrix}, \begin{bmatrix} \overline{a}q_3\overline{\#}\overline{a}\,\overline{q_0} \\ 2 \end{bmatrix}, \begin{bmatrix} \overline{b}q_3\overline{\#}\overline{a}\,\overline{q_0} \\ 2 \end{bmatrix},$$

$$\left. \begin{bmatrix} \overline{a}q_3\overline{\#}\overline{b}\,\overline{q_0} \\ 0 \end{bmatrix}, \begin{bmatrix} \overline{b}q_3\overline{\#}\overline{b}\,\overline{q_0} \\ 0 \end{bmatrix}, \begin{bmatrix} \overline{a}q_1\overline{\#}\overline{a}\,\overline{q_0} \\ 1 \end{bmatrix}, \begin{bmatrix} \overline{b}q_1\overline{\#}\overline{a}\,\overline{q_0} \\ 1 \end{bmatrix}, \begin{bmatrix} \overline{a}q_0\overline{\#}\overline{q_4} \\ 0 \end{bmatrix}, \begin{bmatrix} \overline{b}q_0\overline{\#}\overline{q_4} \\ 0 \end{bmatrix} \right\}.$$

Let us consider a word starting with *abab* and how the weighted word game follows. Starting from the initial element $(q_3\#, 0)$, which represents the final state q_3 of $\mathcal{A}$ and weight zero, Defender plays $(a, 0), (b, 0), (a, 0)$ and $(b, 0)$, while Attacker plays $(\#, 0)$ thrice until Attacker plays $(\overline{b}\,\overline{q_2}, 0)$ to start the simulation of the automaton:

$$\begin{bmatrix} q_3\# \\ 0 \end{bmatrix} \cdot \underset{D}{\begin{bmatrix} a \\ 0 \end{bmatrix}} \underset{A}{\begin{bmatrix} \# \\ 0 \end{bmatrix}} \underset{D}{\begin{bmatrix} b \\ 0 \end{bmatrix}} \underset{A}{\begin{bmatrix} \# \\ 0 \end{bmatrix}} \underset{D}{\begin{bmatrix} a \\ 0 \end{bmatrix}} \underset{A}{\begin{bmatrix} \# \\ 0 \end{bmatrix}} \underset{D}{\begin{bmatrix} b \\ 0 \end{bmatrix}} \underset{A}{\begin{bmatrix} \overline{b}\,\overline{q_2} \\ 0 \end{bmatrix}}.$$

After this moment of the play, it does not matter which letter Defender plays as Attacker can always cancel it. Let $c_1, c_2 \in \{a, b\}$ be the letters Defender plays. Now Attacker

follows the computation path visiting q_1, q_3, q_2 and ending in q_3 in the reverse order:

$$\begin{bmatrix} q_3\# \\ 0 \end{bmatrix} \cdot \underset{D}{\begin{bmatrix} a \\ 0 \end{bmatrix}} \underset{A}{\begin{bmatrix} \# \\ 0 \end{bmatrix}} \underset{D}{\begin{bmatrix} b \\ 0 \end{bmatrix}} \underset{A}{\begin{bmatrix} \# \\ 0 \end{bmatrix}} \underset{D}{\begin{bmatrix} a \\ 0 \end{bmatrix}} \underset{A}{\begin{bmatrix} \# \\ 0 \end{bmatrix}} \underset{D}{\begin{bmatrix} b \\ 0 \end{bmatrix}} \underset{A}{\begin{bmatrix} \overline{b}\,\overline{q_2} \\ 0 \end{bmatrix}} \underset{D}{\begin{bmatrix} a \\ 0 \end{bmatrix}} \underset{A}{\begin{bmatrix} \overline{a}\,q_2\overline{\#}\,\overline{a}\,\overline{q_3} \\ 0 \end{bmatrix}} = \begin{bmatrix} q_3\# \cdot a \cdot \# \cdot b \cdot \# \cdot \overline{q_3} \\ 0 \end{bmatrix}$$

$$\longrightarrow \begin{bmatrix} q_3\# \cdot a \cdot \# \cdot b \cdot \# \cdot \overline{q_3} \\ 0 \end{bmatrix} \underset{D}{\begin{bmatrix} c_1 \\ 0 \end{bmatrix}} \underset{A}{\begin{bmatrix} \overline{c_1}\,q_3\overline{\#}\,\overline{b}\,\overline{q_1} \\ -1 \end{bmatrix}} = \begin{bmatrix} q_3\# \cdot a \cdot \# \cdot \overline{q_1} \\ -1 \end{bmatrix}$$

$$\longrightarrow \begin{bmatrix} q_3\# \cdot a \cdot \# \cdot \overline{q_1} \\ -1 \end{bmatrix} \underset{D}{\begin{bmatrix} c_2 \\ 0 \end{bmatrix}} \underset{A}{\begin{bmatrix} \overline{c_2}\,q_1\overline{\#}\,\overline{a}\,\overline{q_0} \\ 1 \end{bmatrix}} = \begin{bmatrix} q_3\# \cdot \overline{q_0} \\ -1+1 \end{bmatrix}.$$

As the weight of this play is 0, Attacker wins the game by playing the correct $(\overline{c_3}q_0\overline{\#}\overline{q_3}, 0)$.

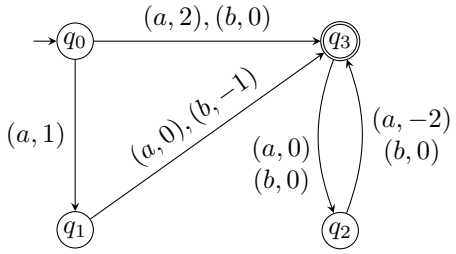

Figure 5: Weighted automaton $\mathcal{A}$.

4.2. Word games on pairs of group words

We now modify the game of the previous subsection by encoding the weight as a separate word over a group alphabet F'_Γ. This encoding, with additional tricks, allows us to construct a word game, where both the initial and final positions are $(\varepsilon, \varepsilon)$.

This variant of the *word game* consists of Attacker and Defender having sets of pair of words

$$\{(u_1, u'_1), \ldots, (u_r, u'_r)\} \subseteq F_\Gamma \times F_{\Gamma'} \quad \text{and} \quad \{(v_1, \varepsilon), \ldots, (v_s, \varepsilon)\} \subseteq F_\Gamma \times F_{\Gamma'}$$

respectively, where Γ and Γ' are binary group alphabets. A configuration of a game after t rounds is an element $w_t = (v_{i_1}, \varepsilon) \cdot (u_{i_1}, u'_{i_1}) \cdot (v_{i_2}, \varepsilon) \cdot (u_{i_2}, u'_{i_2}) \cdot \ldots \cdot (v_{i_t}, \varepsilon) \cdot (u_{i_t}, u'_{i_t})$, where (u_{i_j}, u'_{i_j}) and (v_{i_j}, ε) are elements from above defined sets of Attacker and Defender. The initial position is an element (w, ε) and the target position of this game is the identity element $(\varepsilon, \varepsilon)$. The decision problem for the word game is to check whether there exists a winning strategy for Attacker to reach the identity element $(\varepsilon, \varepsilon)$.

Undecidability of existence of a winning strategy in the word game where elements are generated by $\Gamma \times \{\rho, \overline{\rho}\}$ follows from Theorem 7. This follows from the fact that the free group with one generator is the same thing as the integers $(\mathbb{Z}, +)$. Indeed, consider a move (w, z) of a player in the weighted word game G. In the word game on pair of words, the same player has the move (w, ρ^z). It is easy to see, that the same player has a winning strategy in both games. In the next theorem, we construct a word game, where

both the initial and target elements are $(\varepsilon, \varepsilon)$. We construct a game where Attacker has four copies of moves of G, each encoded over a disjoint group alphabet. We use the idea of the encoding of [34] to ensure that there are four plays of G played in a particular order and $(\varepsilon, \varepsilon)$ is reached in the new game if and only if $(\varepsilon, \varepsilon)$ is reached in all four plays of G.

Theorem 9. *It is undecidable whether Attacker has a winning strategy to reach $(\varepsilon, \varepsilon)$ from $(\varepsilon, \varepsilon)$ in the word game where elements are generated by $\Gamma \times \Gamma'$, where both Γ and Γ' are binary group alphabets, i.e., $\Gamma = \{c, d, \overline{c}, \overline{d}\}$ and $\Gamma' = \{e, f, \overline{e}, \overline{f}\}$.*

Proof. Let G be the word game of Theorem 7 for which deciding whether Attacker has a winning strategy is undecidable. Let U and V be moves of Attacker and Defender of G, where the first component of the move is over a binary alphabet and the second component is over the unary alphabet, and the initial position is (w, ε). Consider the weighted automaton $\mathcal{B}$ of Lemma 6 with a single final state, $Q_{\mathcal{B}}$ is the set of states and A is its input alphabet.

We want to make sure that the four consecutive plays will be played one after another. For this we introduce eight border letters $\square_1, \square_2, \square_3, \square_4, \Diamond_1, \Diamond_2, \Diamond_3, \Diamond_4$ from a fresh group alphabet. Attacker's first component consists of words over the group alphabet $\Gamma_1 = A \cup A^{-1} \cup \{\#, \overline{\#}\} \bigcup_{k=1}^{4} (Q_{\mathcal{B}_k} \cup Q_{\mathcal{B}_k}^{-1})$ and the second component is over the group alphabet $\{\square_k, \rho_k, \Diamond_k, \overline{\square_k}, \overline{\rho_k}, \overline{\Diamond_k} \mid k \in \{1, 2, 3, 4\}\}$. We construct Attacker's set of moves U' which encode the original moves in U over the four alphabets in the following manner:

- Dummy move: $(\#, \varepsilon) \in U$ is added to U' as it is;
- Initialization moves: for each move $(\overline{a}\,\overline{q_i}, \rho^z) \in U$, we add moves $(\overline{a}\,\overline{q_{i1}}, \square_1 \rho_1^z \Diamond_1)$, $(\overline{a}\,\overline{q_{i2}}, \overline{\square_2} \rho_2^z \Diamond_2)$, $(\overline{a}\,\overline{q_{i3}}, \overline{\square_3} \rho_3^z \Diamond_3)$, $(\overline{a}\,\overline{q_{i4}}, \overline{\square_4} \rho_4^z \Diamond_4)$ to U';
- Simulation moves: for each move $(\overline{b} q_i \overline{\#} \overline{a}\,\overline{q_j}, \rho^z) \in U$, where q_j is not the initial state of $\mathcal{B}$, we add moves $(\overline{b} q_{i_k} \overline{\#} \overline{a}\,\overline{q_{j_k}}, \overline{\Diamond_k} \rho_k^z \Diamond_k)$, for each $k \in \{1, 2, 3, 4\}$, to U' ;
- Finishing moves: for each move $(\overline{b} q_i \overline{\#} \overline{a}\,\overline{q_j}, \rho^z) \in U$, where q_j is the initial state of $\mathcal{B}$, we add moves $(\overline{b} q_{i_k} \overline{\#} \overline{a}\,\overline{q_{j_k}}, \overline{\Diamond_k} \rho_k^z \square_{k+1})$, for each $k \in \{1, 2, 3\}$, and a move $(\overline{b} q_{i_4} \overline{\#} \overline{a}\,\overline{q_{j_4}}, \overline{\Diamond_4} \rho_k^z \overline{\square_1})$ to U';
- Finally, to finish the game, we add $(\overline{a} q_{j_4} q_{j_3} q_{j_2} q_{j_1}, \varepsilon)$, where q_j is the initial state of $\mathcal{B}$, to U'.

From the way how the moves are constructed, it follows that the only way to cancel all the border letters (i.e., $\square_i, \Diamond_i$ and q_{j_i} for $i = 1, \ldots, 4$), is to have four consecutive plays of game G followed by the move $(\overline{a} q_{j_4} q_{j_3} q_{j_2} q_{j_1}, \varepsilon)$. Namely, first using the moves containing letters from the alphabet $Q_{\mathcal{B}_1}$, then $Q_{\mathcal{B}_2}$, followed by $Q_{\mathcal{B}_3}$ and finally $Q_{\mathcal{B}_4}$, or a cyclic permutation of the order. If the moves are played in a different order, then the border letters will create non-cancellable pair of elements from two distinct alphabets. From the way the moves are constructed, it is impossible to reach $(\varepsilon, \varepsilon)$ afterwards.

Then, if Attacker has a winning strategy to reach $(\varepsilon, \varepsilon)$ in the weighted word game G, she also has a winning strategy to reach $(\varepsilon, \varepsilon)$ in the word game on a pair of words. On the other hand, if Defender has a winning strategy in the weighted word game, then no matter how Attacker plays each of the four plays, either the first or the second component will remain non-empty (or both). After the whole cycle is played, in the two components,

there will be letters over at least four distinct alphabets. Since Attacker does not have a winning strategy in G and due to the usage of the border letters, no matter how Attacker will play a second cycle, the number of distinct alphabets will not decrease. So the identity element $(\varepsilon, \varepsilon)$ cannot be generated by a concatenation of four plays of G unless, Attacker has a winning strategy in G. A similar idea of encoding generators over four alphabets has been used in [34].

Finally, we encode the words in both components using the monomorphism of Lemma 5 to have the game over binary group alphabets Γ and Γ'. □

Example 10. Consider the weighted word game G of Example 8 from which we construct the set U' as in the previous theorem. To illustrate the idea of the encoding, let us consider a prefix of a play

$$\underset{D}{\begin{bmatrix} a \\ \varepsilon \end{bmatrix}} \cdot \underset{A}{\begin{bmatrix} \# \\ \varepsilon \end{bmatrix}} \cdot \underset{D}{\begin{bmatrix} b \\ \varepsilon \end{bmatrix}} \cdot \underset{A}{\begin{bmatrix} \overline{b}\,\overline{q_{11}} \\ \square_1 \rho_1^{-1} \Diamond_1 \end{bmatrix}} \cdot \underset{D}{\begin{bmatrix} a \\ \varepsilon \end{bmatrix}} \cdot \underset{A}{\begin{bmatrix} \overline{a} q_{11} \overline{\#} \overline{a}\,\overline{q_{01}} \\ \overline{\Diamond_1} \rho_1 \square_2 \end{bmatrix}} \cdot \underset{D}{\begin{bmatrix} a \\ \varepsilon \end{bmatrix}},$$

where moves of Defender are indicated with the letter D and moves of Attacker with the letter A. The reduced element of this prefix is $(\overline{q_{01}}a, \square_1\square_2)$. The only moves that cancel $\square_2$ in the second component have letters over $\mathcal{B}_2$ in the first component. That is, a new play of G is simulated using the second alphabet.

4.3. Matrix games on vectors

We extend the domain of the game and a set of rules to the class of linear transformations on integer lattice $\mathbb{Z}^4$. A *matrix game on vectors* (or a *matrix game* for short) consists of two players, Attacker and Defender, having sets of linear transformations $\{U_1, \ldots, U_r\} \subseteq \mathbb{Z}^{n\times n}$ and $\{V_1, \ldots, V_s\} \subseteq \mathbb{Z}^{n\times n}$ respectively, an *initial vector* $\mathbf{x}_0 \in \mathbb{Z}^n$ of the game representing the starting position, and a *target vector* $\mathbf{y} \in \mathbb{Z}^n$. The *dimension* of the game is the dimension of the integer lattice n. Starting from $\mathbf{x}_0$, players move the current point by applying available linear transformations (by matrix multiplication) from their respective sets in turns. The decision problem of the matrix game is to check whether there exists a winning strategy for Attacker to reach the target from the starting point (vectors in $\mathbb{Z}^n$) of the game. Note that in our formulation the vectors are horizontal and players multiply them from the right. Recall that $SL(n, \mathbb{Z}) = \{M \in \mathbb{Z}^{n\times n} \mid \det(M) = 1\}$.

In a game where each player has only one possible move (i.e., when the game is deterministic), the existence of a winning strategy for Attacker or Defender can be trivially reduced to the orbit problem by combining the two matrices into one. The *orbit problem* is decidable in polynomial time for any matrix size over integers, rationals and even algebraic numbers [35].

If Defender has a single move, the problem of checking whether Attacker has a winning strategy corresponds to the standard vector reachability problem which has been extensively studied in the literature [36, 37, 38, 39, 40, 41, 42]. Indeed, let $A = \{U_1, \ldots, U_k\}$ be a matrix set for which the vector reachability problem is undecidable (for vectors $\mathbf{x}, \mathbf{y}$) and $B = \{V\}$ ($V \in SL(n, \mathbb{Z})$). Then let us consider the matrix game with Attacker's set $\{V^{-1}U_1, \ldots, V^{-1}U_k\}$, Defender's set B, the initial position $\mathbf{x}$, and the target $\mathbf{y}$. In this game, Attacker has a winning strategy if and only if $\mathbf{y}$ is reachable from $\mathbf{x}$ in the vector reachability problem. Thus, a game where Attacker has 6 matrices in $\mathbb{Z}^{3\times 3}$, and Defender has a single matrix from $SL(3, \mathbb{Z})$ is undecidable, following the

undecidability of the vector reachability problem for 6 integer matrices in dimension three [40].

In the symmetric case when Attacker has a single matrix and Defender has m matrices, we can reduce the problem of checking for the existence of a winning strategy to a different reachability problem for matrix semigroups with a stronger reachability objective. That is, Attacker's set is $A = \{U\}$, Defender's set is $\{V_1, \ldots, V_m\}$, the initial position $\mathbf{x}$, and the target position $\mathbf{y}$. Again we can combine sets A and B into one generating set $B' = \{V_1U, \ldots, V_mU\}$ of a semigroup S. However, the question whether Attacker has a winning strategy would require for us to check that on any infinite trajectory of reachable points starting from the initial point $\mathbf{x}$ and transformed by elements of S, the point $\mathbf{y}$ is eventually reachable. Let us now formally define this problem.

Problem 11. *Let S be a matrix semigroup generated by $B = \{V_1, \ldots, V_m\}$ and $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^n$. In the* eventual reachability problem*, we are asked whether for every element $M = V_{i_1} \cdots V_{i_k}$ either there exists $j \leq k$ such that $\mathbf{x}V_{i_1} \ldots V_{i_j} = \mathbf{y}$ or there exists $N \in S$ such that $\mathbf{x}MN = \mathbf{y}$. In other words, $\mathbf{y}$ appears in every trajectory starting from $\mathbf{x}$.*

To the authors best knowledge, this problem has not been studied previously. The problem is illustrated in Figure 6. The solution of this problem gives the answer to whether a winning strategy exists in a matrix game where Attacker has one move and Defender has several moves.

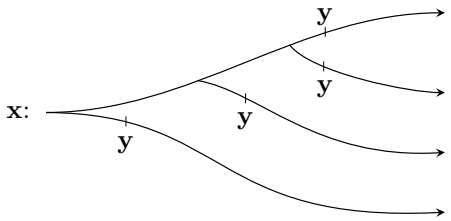

Figure 6: An illustration of traces in the eventual reachability problem.

Let us consider a case where both Attacker and Defender have at least two moves. Let us assume that $B = \{V_1, V_2\} \subseteq SL(n, \mathbb{Z})$ and $A = \{U_1, U_2\}$. Consider a game where Defender's set is B and Attacker's set is

$$A' = \{U_1V_1^{-1}, U_1V_2^{-1}, U_2V_1^{-1}, U_2V_2^{-1}\}.$$

The previous reasoning of reducing the problem to the standard vector reachability for matrix semigroups is not directly applicable. The reachability of $\mathbf{y}$ from $\mathbf{x}$ using matrices from A implies that there exists a winning strategy in the matrix game. Attacker's strategy is to follow the solution of the reachability problem with the moves that cancel the matrix played by Defender (i.e., if Defender played V_i, then Attacker plays $U_jV_i^{-1}$). On the other hand, the existence of a winning strategy does not imply that $\mathbf{y}$ is reachable from $\mathbf{x}$. Indeed, unlike previously, Attacker might follow V_1 with $V_2^{-1}U_1$ and can still reach the target position. If we try to consider the reachability using matrices from A', containing all the possible moves of Defender followed by all the possible moves of Attacker, then from the fact that $\mathbf{y}$ is reachable from $\mathbf{x}$, it would not follow that Attacker

has a winning strategy. Indeed, this kind of trick eliminates the role of Defender and says very little in relation to the game.

Next we prove the main theorem regarding the matrix game, where both Attacker and Defender have at least two moves.

Theorem 12. *Given two finite sets of matrices* $\{U_1, U_2, \ldots, U_r\} \subseteq \mathbb{Z}^{n\times n}$ *for* Attacker *and* $\{V_1, V_2, \ldots, V_s\} \subseteq \mathbb{Z}^{n\times n}$ *for* Defender*, where* $r, s \geq 2$*, an initial starting vector* $\mathbf{x}_0 \in \mathbb{Z}^n$ *and a target vector* $\mathbf{y} \in \mathbb{Z}^n$*, it is undecidable whether Attacker has a winning strategy in the matrix game. Furthermore, the claim holds even when the matrices are from* $SL(4, \mathbb{Z})$*.*

Proof. Let $\Sigma_2 = \{c, d, \bar{c}, \bar{d}\}$ be a binary group alphabet and define $f : \Sigma_2^* \to SL(2, \mathbb{Z})$ by: $f(c) = \left(\begin{smallmatrix} 1 & 2 \\ 0 & 1 \end{smallmatrix}\right), f(\bar{c}) = \left(\begin{smallmatrix} 1 & -2 \\ 0 & 1 \end{smallmatrix}\right), f(d) = \left(\begin{smallmatrix} 1 & 0 \\ 2 & 1 \end{smallmatrix}\right), f(\bar{d}) = \left(\begin{smallmatrix} 1 & 0 \\ -2 & 1 \end{smallmatrix}\right)$.

Then mapping f is a monomorphism [33] and $f(\varepsilon)$ corresponds to the identity matrix in $\mathbb{Z}^{2\times 2}$. Let α be a function defined in Lemma 5, then by the following straightforward matrix multiplication we have:

$$f(\alpha(z_j)) = f(c^j d \bar{c}^j) = \begin{pmatrix} 1+4j & -8j^2 \\ 2 & 1-4j \end{pmatrix}.$$

Let us show that if $(1,0)M = (1,0)$, where M is an image of a word over binary group alphabet under f, that is, $M \in \{f(\alpha(w)) \mid w \in F_\Gamma\}$, then M is the identity matrix. The reasoning follows [33]. Let $M = \left(\begin{smallmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{smallmatrix}\right)$, now $(1,0)M = (m_{11}, m_{12})$ which implies that $m_{11} = 1$ and $m_{12} = 0$. By the previous observation, $m_{22} = 1$. The final letter of $\alpha(w)$ is $\bar{c}$, which is $\left(\begin{smallmatrix} 1 & -2 \\ 0 & 1 \end{smallmatrix}\right)$ under f. Let $Y = f(\alpha(w))f(\bar{c})^{-1} = \left(\begin{smallmatrix} x & y \\ z & v \end{smallmatrix}\right)$. Now $f(\alpha(w)) = \left(\begin{smallmatrix} x & y \\ z & v \end{smallmatrix}\right)\left(\begin{smallmatrix} 1 & -2 \\ 0 & 1 \end{smallmatrix}\right) = \left(\begin{smallmatrix} x & y-2x \\ z & v-2z \end{smallmatrix}\right)$. Since $x = 1$, $y - 2x = 0$ and $x - 2x = 1$, we see that $f(\alpha(w)) = \left(\begin{smallmatrix} 1 & 0 \\ z & 1 \end{smallmatrix}\right) = f(d)^{z/2}$ but by the definition of the encoding this is possible only when $z = 0$. This implies that $f(\alpha(w))$ is the identity matrix.

Let us encode the word game into the matrix game. Recall that by Theorem 9, it is undecidable whether Attacker has a winning strategy to reach $(\varepsilon, \varepsilon)$ from $(\varepsilon, \varepsilon)$ in a word game. We construct 4×4 matrices with words of the first component encoded by f in the upper left corner and words from the second components encoded by f in the lower right corner. The direct application of the above function to the elements of the word game gives us a set of matrices for Attacker and a set of matrices for Defender from $SL(4, \mathbb{Z})$. By previous considerations for vector $x_0 = (1, 0, 1, 0)$, the equation $x_0 = x_0 \cdot M$, where $M \in SL(4, \mathbb{Z})$ has only one matrix M satisfying the above statement, the identity matrix in $\mathbb{Z}^{4\times 4}$. Therefore, for every matrix game, where the initial vector x_0 is $(1, 0, 1, 0)$, the question about the winning strategy of reaching x_0 is equivalent to the question of reaching the identity matrix in the product with alternation in applications of Defender's and Attacker's linear transformations, which in its turn corresponds to reaching the identity element in the word game. □

Example 13. Consider a matrix game where Defender has matrices $\left(\begin{smallmatrix} 3 & 5 \\ 1 & 2 \end{smallmatrix}\right)$ and $\left(\begin{smallmatrix} -1 & -1 \\ 0 & -1 \end{smallmatrix}\right)$ and Attacker has $\left(\begin{smallmatrix} 1 & -2 \\ -2 & 3 \end{smallmatrix}\right)$, $\left(\begin{smallmatrix} 1 & -2 \\ 3 & -7 \end{smallmatrix}\right)$, $\left(\begin{smallmatrix} 7 & -17 \\ -5 & 12 \end{smallmatrix}\right)$, $\left(\begin{smallmatrix} 1 & -2 \\ 3 & -7 \end{smallmatrix}\right)$.

Consider a play from $\mathbf{x}_0 = (1, 0)$ where Defender plays $M_1 = \left(\begin{smallmatrix} 3 & 5 \\ 1 & 2 \end{smallmatrix}\right)$ followed by $M_3 = \left(\begin{smallmatrix} -1 & -1 \\ 0 & -1 \end{smallmatrix}\right)$ and Attacker plays $M_2 = \left(\begin{smallmatrix} 1 & -2 \\ -2 & 3 \end{smallmatrix}\right)$ followed by $M_4 = \left(\begin{smallmatrix} 1 & -2 \\ 3 & -7 \end{smallmatrix}\right)$.

$$(1,0)\begin{pmatrix} 3 & 5 \\ 1 & 2 \end{pmatrix}\begin{pmatrix} 1 & -2 \\ -2 & 3 \end{pmatrix}\begin{pmatrix} -1 & -1 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} 1 & -2 \\ 3 & -7 \end{pmatrix} = (1,0).$$

From this computation, we see that the play is winning for Attacker. With similar computations we can see that in fact Attacker has a winning strategy in this game. The play $\mathbf{x}_0 M_1 M_2 M_3 M_4$ is depicted in Figure 7.

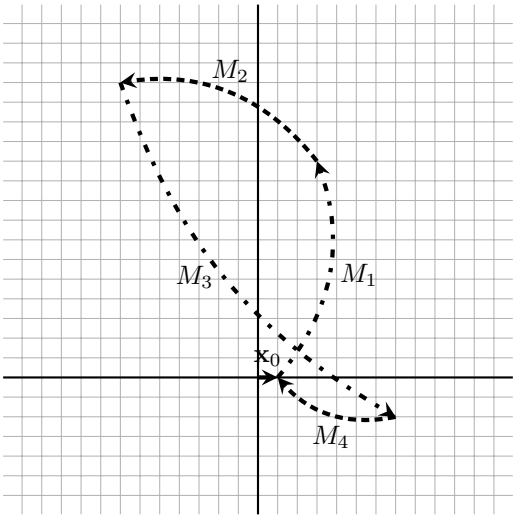

Figure 7: The play $\mathbf{x}_0 M_1 M_2 M_3 M_4$ of a matrix game.

4.4. Braid games

In this subsection we consider the Attacker-Defender games on topological objects, braids in B_n. The moves of the game are compositions of braids in B_3 (a class of braids with only three strands) and B_5 (a class of braids with only five strands) [21]. Braids are classical topological objects that attracted a lot of attention due to their connections to topological knots and links, as well as their applications to polymer chemistry, molecular biology, cryptography, quantum computations and robotics [16, 17, 18, 19, 20]. There is also recent interest about the complexity and termination of the games on braids [14, 15] that are defined with specific rules of adding and removing crossings. In this paper we consider very simple games on braids with only three or five strands (i.e., B_3 or B_5) where the braid is modified by a composition with a finite set of braids. We show that it is undecidable to check the existence of a winning strategy in such games, while the reachability with a single player (i.e., with nondeterministic composition from a single set) was shown to be decidable for B_3 and undecidable for B_5 in [21].

Definition 1. The n-strand braid group B_n is the group given by the presentation with $n-1$ generators $\sigma_1, \ldots, \sigma_{n-1}$ and the following relations $\sigma_i \sigma_j = \sigma_j \sigma_i$, for $|i-j| \geq 2$ and $\sigma_i \sigma_{i+1} \sigma_i = \sigma_{i+1} \sigma_i \sigma_{i+1}$ for $1 \leq i \leq n-2$. These relations are called Artin's relations.

Elements of the braid group B_n will be represented by words in the alphabet

$$\{\sigma_1, \ldots, \sigma_{n-1}, \sigma_1^{-1}, \ldots, \sigma_{n-1}^{-1}\}$$

and we refer to them as braid words[4].

A composition of two braids with four strands is illustrated in Figure 8.
The *fundamental braid* of B_n is

$$\Delta_n = (\sigma_{n-1}\sigma_{n-2}\dots\sigma_1)(\sigma_{n-1}\sigma_{n-2}\dots\sigma_2)\dots\sigma_{n-1}.$$

Geometrically, the fundamental braid is obtained by lifting the bottom ends of the identity braid and flipping (right side over left) while keeping the ends of the strings in a line.

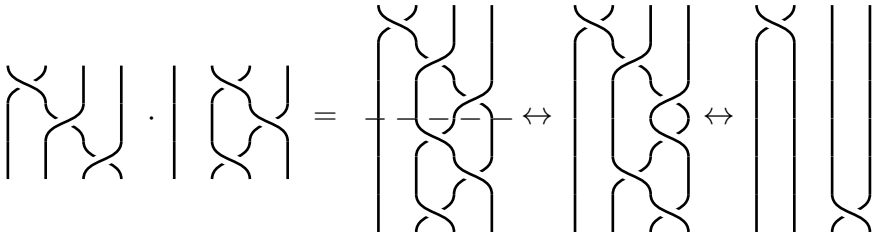

Figure 8: An example of a composition of braids in B_4.

The braid game can be defined in a way, where the sets of braid words $\{a_1, ..., a_r\}$ and $\{d_1, ..., d_s\}$, for Attacker and Defender respectively, will correspond to braids in B_n. The initial braid of the game is given and each following configuration of the game is changed by Attacker or Defender by composing braids from their corresponding sets. Given two geometric braids, we can compose them, i.e., put one after the other making the endpoints of the first one coincide with the starting points of the second one. There is a neutral element for the composition: it is the trivial braid, also called the identity braid, i.e., the class of the geometric braids where all the strings are straight. Two geometric braids are isotopic if there is a continuous deformation of the ambient space that deforms one into the other, by a deformation that keeps every point in the two bordering planes fixed.

Finally, the goal of Attacker is to unbraid, i.e., to reach a configuration of the game that is isotopic to the trivial braid (empty word) and Defender tries to keep Attacker from reaching it. Two braids are isotopic if their braid words can be translated one into each other via the relations from Definition 1 plus the relations $\sigma_i\sigma_i^{-1} = \sigma_i^{-1}\sigma_i = 1$ where 1 is the identity (trivial braid).

Theorem 14. *The braid game is undecidable for braids from B_3 starting from a non-trivial braid and for braids from B_5 starting from the trivial braid.*

Proof. We encode the undecidable weighted word game of Theorem 7 into a braid game with three strands and the undecidable word game of Theorem 9 into a braid game with five strands and show that the respective braid games are undecidable as well.

[4]Whenever a crossing of strands i and $i+1$ is encountered, either σ_i or ${\sigma_i}^{-1}$ is written down, depending on whether the strand i moves under or over the strand $i+1$.

Let $\Sigma_2 = \{c, d, \overline{c}, \overline{d}\}$ be a binary group alphabet and define $f : \Sigma_2^* \to B_3$ by: $f(c) = {\sigma_1}^4, f(\overline{c}) = {\sigma_1}^{-4}, f(d) = {\sigma_2}^4, f(\overline{d}) = {\sigma_2}^{-4}$. Then mapping f is a monomorphism [43]. Let α be the mapping from Lemma 5 then:

$$f(\alpha(z_j)) = f(c^j d \overline{c}^j) = {\sigma_1}^{4j} {\sigma_2}^4 {\sigma_1}^{-4j}$$

and the length of a braid word from B_3 corresponding to a letter $z_j \in \Sigma'$ is $8j + 4$. The above morphisms give a way to map words from an arbitrary sized group alphabet into the set of braid words in B_3.

Now we again can use the weighted word game as any word over a binary group alphabet can be uniquely mapped into a braid, where the empty word will correspond to a braid which is isotopic to the trivial braid and the concatenation of words over group alphabet corresponds to the composition of braids in B_3. The weight $x \in \mathbb{Z}$ is mapped into the braid word Δ_3^{2x} where $\Delta_3^2 = (\sigma_1\sigma_2\sigma_1)^2$ is a central element of B_3.

Subgroups $\langle \sigma_1^4, \sigma_2^4 \rangle, \langle \sigma_4^2, d \rangle$ of the group B_5 are free and B_5 contains the direct product $\langle \sigma_1^4, \sigma_2^4 \rangle \times \langle \sigma_4^2, d \rangle$ of two free groups of rank 2 as a subgroup where $d = \sigma_4\sigma_3\sigma_2\sigma_1^2\sigma_2\sigma_3\sigma_4$ [43]. Now we can uniquely encode pair of words of the word game into B_5. Using the word game, where the initial word is $(\varepsilon, \varepsilon)$, we can construct a braid game from B_5 starting from the trivial braid.

It is easy to see that Attacker has a winning strategy in a braid game on B_3 and B_5 if and only if she has a winning strategy in a weighted word game and a word game on pairs of group words, respectively. □

Example 15. Consider a braid game on B_3 where Defender has braid words $\sigma_1\sigma_2^{-1}$ and $\sigma_1^{-1}\sigma_2^{-1}\sigma_1^{-1}$ and Attacker has braid words $\sigma_2\sigma_1\sigma_2$ and $\sigma_2\sigma_1$. Starting from $\sigma_1^{-1}\sigma_1^{-1}$, we have the following play:

$$\begin{aligned}\left[\sigma_1^{-1}\sigma_1^{-1}\right] \cdot \underset{D}{\left[\sigma_1\sigma_2^{-1}\right]} \cdot \underset{A}{\left[\sigma_2\sigma_1\sigma_2\right]} \cdot \underset{D}{\left[\sigma_1^{-1}\sigma_2^{-1}\sigma_1^{-1}\right]} \cdot \underset{A}{\left[\sigma_2\sigma_1\right]} &= \left[\sigma_2\sigma_1^{-1}\sigma_2^{-1}\sigma_1^{-1}\sigma_2\sigma_1\right] \\ &= \left[\sigma_2\sigma_2^{-1}\sigma_1^{-1}\sigma_2^{-1}\sigma_2\sigma_1\right] = [1]\,,\end{aligned}$$

where the second equality follows from the relation $\sigma_1\sigma_2\sigma_1 = \sigma_2\sigma_1\sigma_2$ of Definition 1. This play is depicted in Figure 9.

From the definition of braids, braids with two strands represent integers with a braid word σ_1^z corresponding to an integer $z \in \mathbb{Z}$. From the decidability of robot games in dimension one [1], it follows that the braid game on B_2 is also decidable. The braid game on B_3 is the first non-trivial case that is undecidable. In B_5, the game starting from the trivial braid was shown to be undecidable. The status of the braid games on B_3 and B_4 starting from the trivial braid are open.

5. Conclusions

The results of the paper are twofold. We have proven a new language-theoretic result for weighted automata on infinite words. We constructed an automaton that, for a given instance of the ω PCP, accepts all the infinite words that are not the solutions of the instance of the ω PCP. In other words, the non-universality of the automaton corresponds to the instance of the ω PCP having a solution. Secondly, we have shown how to encode

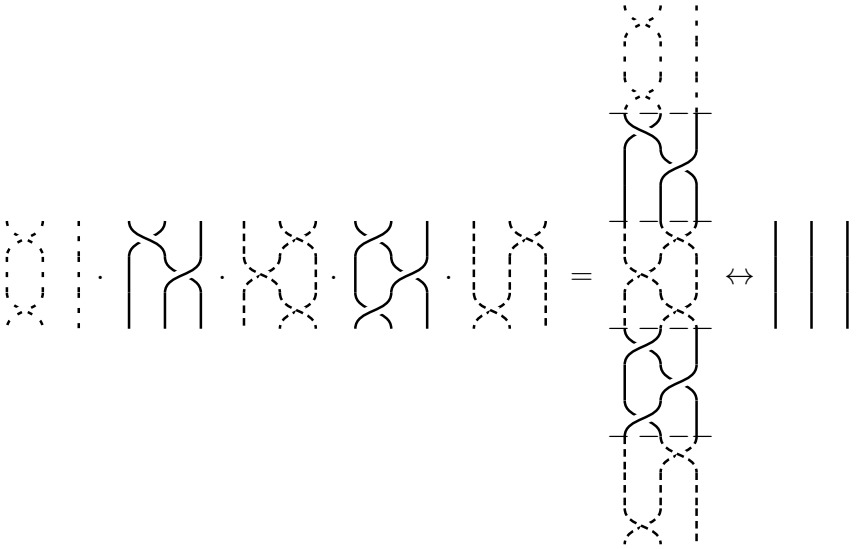

Figure 9: An example of a braid game, where the initial braid is dotted, braids played by Attacker dashed and braids played by Defender in solid.

the automaton into the framework of Attacker-Defender games, from which we obtained undecidability results for checking for the existence of a winning strategy in word games, matrix games on vectors and braid games.

For weighted automata on infinite words, the status of the universality problem remains open for automata with less than five states. For the matrix game on vectors, it is unknown whether deciding the winner is decidable for dimensions two and three. The braid games on B_3 and B_4, where the initial braid is trivial, remain open. However, the direct application of the word game is not applicable due to the fact that there is no faithful representation of the direct product of two free groups of rank two into B_4 [44]. Furthermore, it would be interesting to see whether similar encoding techniques of a weighted automaton are applicable to other games such as robot games with states.

The complexity of Attacker-Defender games considered in this paper is quite sensitive to the process of determinization for one or both players, as it was shown in the case of matrix games in Subsection 4.3. The case where both players have a deterministic single move can be reduced to the well-studied orbit problem which is decidable in polynomial time. The case where Defender has a deterministic move, leads to a reduction to the vector reachability problem which is undecidable starting from dimension three. The opposite case where Attacker has no choice leads to a new question that we call "eventual reachability". Finally, when both Attacker and Defender have more than one move, the game is not directly reducible to previously studied problems and it is shown to be undecidable in general following a new language-theoretic result for weighted automata on infinite words. The above scenarios about determinism in moves have similar effects on other games from this paper and remain a subject for further study.

Acknowledgement

The authors would like to thank the anonymous reviewers and Prof. Jeffrey Shallit for their helpful comments and suggestions.

References

[1] A. Arul, J. Reichert, The Complexity of Robot Games on the Integer Line, in: Proceedings of QAPL 2013, vol. 117 of *EPTCS*, 132–148, doi:10.4204/EPTCS.117.9, 2013.

[2] P. A. Abdulla, A. Bouajjani, J. d'Orso, Deciding Monotonic Games, in: Proceedings of CSL 2003, vol. 2803 of *LNCS*, 1–14, doi:10.1007/978-3-540-45220-1_1, 2003.

[3] T. Brázdil, P. Jančar, A. Kučera, Reachability Games on Extended Vector Addition Systems with States, in: Proceedings of ICALP 2010, vol. 6199 of *LNCS*, 478–489, doi:10.1007/978-3-642-14162-1_40, 2010.

[4] K. Chatterjee, N. Fijalkow, Infinite-state games with finitary conditions, in: Proceedings of CSL 2013, vol. 23 of *LIPIcs*, 181–196, doi:10.4230/LIPIcs.CSL.2013.181, 2013.

[5] L. Doyen, A. Rabinovich, Robot games. Personal website, 2011, Tech. Rep. LSV-13-02, LSV, ENS Cachan, URL `http://www.lsv.ens-cachan.fr/Publis/RAPPORTS%5FLSV/PDF/rr-lsv-2013-02.pdf`, 2013.

[6] O. Kupferman, M. Y. Vardi, P. Wolper, An automata-theoretic approach to branching-time model checking, J. ACM 47 (2) (2000) 312–360, doi:10.1145/333979.333987.

[7] I. Walukiewicz, Pushdown Processes: Games and Model-Checking, Inf. Comput. 164 (2) (2001) 234–263, doi:10.1006/inco.2000.2894.

[8] R. Alur, T. A. Henzinger, O. Kupferman, Alternating-time temporal logic, J. ACM 49 (5) (2002) 672–713, doi:10.1145/585265.585270.

[9] M. O. Rabin, Decidability of second-order theories and automata on infinite trees, Bull. Amer. Math. Soc. 74 (5) (1968) 1025–1029, doi:10.1090/S0002-9947-1969-0246760-1.

[10] J. Reichert, Reachability Games with Counters: Decidability and Algorithms, Doctoral thesis, Laboratoire Spécification et Vérification, ENS Cachan, France, 2015.

[11] M. Kunc, Regular solutions of language inequalities and well quasi-orders, Theor. Comput. Sci. 348 (2-3) (2005) 277–293, doi:10.1016/j.tcs.2005.09.018.

[12] M. Kunc, The Power of Commuting with Finite Sets of Words, Theory Comput. Syst. 40 (4) (2007) 521–551, doi:10.1007/s00224-006-1321-z.

[13] O. Ly, Z. Wu, On effective construction of the greatest solution of language inequality, Theoretical Computer Science 528 (0) (2014) 12 – 31, ISSN 0304-3975, doi:10.1016/j.tcs.2014.02.001.

[14] L. Carlucci, P. Dehornoy, A. Weiermann, Unprovability results involving braids, Proceedings of the London Mathematical Society 102 (1) (2011) 159–192.

[15] A. Bovykin, L. Carlucci, Long games on braids, Preprint. Available online at http://logic.pdmi.ras.ru/~ andrey/braids_final3.pdf .

[16] G. P. Collins, Computing with quantum knots, Sci. Am. 294 (4) (2006) 56–63, doi: 10.1038/scientificamerican0406-56.

[17] P. Dehornoy, I. Dynnikov, D. Rolfsen, B. Wiest, Ordering braids, volume 148 of Mathematical Surveys and Monographs, Amer. Math. Soc., Providence, RI .

[18] D. Epstein, M. Paterson, J. Cannon, D. Holt, S. Levy, W. P. Thurston, Word processing in groups, AK Peters, Ltd., 1992.

[19] D. Garber, Braid group cryptography, Braids: Introductory Lectures on Braids, Configurations and Their Applications 19 (2010) 329.

[20] P. Panangaden, É. O. Paquette, A categorical presentation of quantum computation with anyons, in: New structures for Physics, Springer, 983–1025, 2011.

[21] I. Potapov, Composition Problems for Braids, in: Proceedings of FSTTCS 2013, vol. 24 of *LIPIcs*, 175–187, doi:10.4230/LIPIcs.FSTTCS.2013.175, 2013.

[22] K. Ruohonen, Reversible Machines and Post's Correspondence Problem for Biprefix Morphisms, J. of Information Processing and Cybernetics 21 (12) (1985) 579–595.

[23] V. Halava, T. Harju, Undecidability in Integer Weighted Finite Automata, Fundam. Inform. 38 (1-2) (1999) 189–200, doi:10.3233/FI-1999-381215.

[24] S. Almagor, U. Boker, O. Kupferman, What's Decidable about Weighted Automata?, in: Proceedings of ATVA 2011, vol. 6996 of *LNCS*, 482–491, doi:10.1007/978-3-642-24372-1_37, 2011.

[25] E. L. Post, A variant of a recursively unsolvable problem, Bull. Amer. Math. Soc. 52 (4) (1946) 264–268.
[26] T. Neary, Undecidability in Binary Tag Systems and the Post Correspondence Problem for Five Pairs of Words, in: STACS 2015, vol. 30 of *LIPIcs*, 649–661, doi:10.4230/LIPIcs.STACS.2015.649, 2015.
[27] V. Halava, T. Harju, Undecidability of infinite post correspondence problem for instances of Size 9, ITA 40 (4) (2006) 551–557, doi:10.1051/ita:2006039.
[28] J. Dong, Q. Liu, Undecidability of infinite post correspondence problem for instances of size 8, RAIRO - Theor. Inf. and Applic. 46 (3) (2012) 451–457, doi:10.1051/ita/2012015.
[29] R. Niskanen, I. Potapov, J. Reichert, Undecidability of Two-dimensional Robot Games, in: Proceedings of MFCS 2016, vol. 58 of *LIPIcs*, 73:1–73:13, doi:10.4230/LIPIcs.MFCS.2016.73, 2016.
[30] D. A. Martin, Borel Determinacy, Annals of Mathematics 102 (2) (1975) 363–371, ISSN 0003486X, doi:10.2307/1971035.
[31] E. Grädel, W. Thomas, T. Wilke (Eds.), Automata, Logics, and Infinite Games: A Guide to Current Research, vol. 2500 of *LNCS*, Springer, ISBN 3-540-00388-6, 2002.
[32] J.-C. Birget, S. W. Margolis, Two-letter group codes that preserve aperiodicity of inverse finite automata, in: Semigroup Forum, vol. 76, Springer, 159–168, 2008.
[33] P. C. Bell, I. Potapov, On the Computational Complexity of Matrix Semigroup Problems, Fundam. Inform. 116 (1-4) (2012) 1–13, doi:10.3233/FI-2012-663.
[34] P. C. Bell, I. Potapov, On the Undecidability of the Identity Correspondence Problem and its Applications for Word and Matrix Semigroups, Int. J. Found. Comput. Sci. 21 (6) (2010) 963–978, doi:10.1142/S0129054110007660.
[35] R. Kannan, R. J. Lipton, Polynomial-time Algorithm for the Orbit Problem, J. ACM 33 (4) (1986) 808–821, ISSN 0004-5411, doi:10.1145/6490.6496.
[36] P. Bell, I. Potapov, Reachability problems in quaternion matrix and rotation semigroups, Inf. Comput. 206 (11) (2008) 1353–1361, doi:10.1016/j.ic.2008.06.004.
[37] P. Bell, I. Potapov, On undecidability bounds for matrix decision problems, Theor. Comput. Sci. 391 (1-2) (2008) 3–13, doi:10.1016/j.tcs.2007.10.025.
[38] V. D. Blondel, J. N. Tsitsiklis, A survey of computational complexity results in systems and control, Automatica 36 (9) (2000) 1249–1274, doi:10.1016/S0005-1098(00)00050-9.
[39] S. Gaubert, R. Katz, Reachability Problems for Products of Matrices in Semirings, IJAC 16 (3) (2006) 603–627, doi:10.1142/S021819670600313X.
[40] V. Halava, T. Harju, M. Hirvensalo, Undecidability Bounds for Integer Matrices Using Claus Instances, Int. J. Found. Comput. Sci. 18 (5) (2007) 931–948, doi:10.1142/S0129054107005066.
[41] A. Lisitsa, I. Potapov, Membership and Reachability Problems for Row-Monomial Transformations, in: Proceedings of MFCS 2004, vol. 3153 of *LNCS*, 623–634, doi:10.1007/978-3-540-28629-5_48, 2004.
[42] I. Potapov, From Post Systems to the Reachability Problems for Matrix Semigroups and Multicounter Automata, in: Proceedings of DLT 2004, vol. 3340 of *LNCS*, 345–356, doi:10.1007/978-3-540-30550-7_29, 2004.
[43] V. N. Bezverkhnii, I. V. Dobrynina, Undecidability of the conjugacy problem for subgroups in the colored braid group R_5, Math. Notes 65 (1) (1999) 15–22, doi:10.1007/BF02675004.
[44] A. Akimenkov, Subgroups of the braid group B_4, Mathematical Notes 50 (6) (1991) 1211–1218.